Libreria LPC845

Generado por Doxygen 1.8.13

Índice general

Capítulo 1

Pagina principal de la documentacion

1.1. Introduccion

Esta libreria esta pensada para... (bla bla bla)

Mañana mismo me pongo a escribir esto!

Capítulo 2

Indice de módulos

2.1. Módulos

Lista de todos los módulos:

Capítulo 3

Índice de estructura de datos

3.1. Estructura de datos

Lista de estructuras con una breve descripción:

Capítulo 4

Indice de archivos

4.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 5

Documentación de módulos

5.1. ADC

5.1.1. Descripción detallada

Descripción

Este periférico como su nombre lo indica, convierte una o más entradas analógicas, a un valor equivalente digital. En el caso del LPC845, tiene un único módulo *ADC* con una resolución de 12 bits, el cual tiene 12 canales, lo cual implica que se pueden realizar conversiones de 12 fuentes analógicas distintas, pero no así realizar conversiones *al mismo tiempo* . En caso de querer tomar señales de múltiples fuentes analógicas, se deberán hacer sucesivas conversiones en los distintos canales deseados.

Una resolución de 12 bits implica que la conversión aumentará cada unidad siguiendo la siguiente ecuación↩ : $ADC_{res} = \frac{V_{ref_p}}{2^N}$

Esto implica que podemos preveer el valor resultante de la conversión analógica/digital mediante la siguiente ecuación: $ADC_{conv} = \frac{V_{ADC_{in}}}{ADC_{res}}$

Cabe destacar, que las conversiones serán redondeadas **siempre** hacia abajo, es decir, se descartan los valores decimales.

Concepto de *Secuencia de conversión*

Para el *ADC* de este microcontrolador, un inicio de conversión en realidad puede implicar el inicio de una *secuencia de conversión*. Dicha secuencia puede implicar uno o más canales a convertir, y puede generar eventos tanto cuando se termina la secuencia completa, o cuando se termina cada canal de la secuencia. Asimismo los inicios de conversión pueden disparar una secuencia completa, o el próximo de los canales de dicha secuencia. Se tienen dos secuencias configurables (*Secuencia A y Secuencia B*), las cuales se pueden configurar de forma tal que una secuencia interrumpa a la otra.

Inicio de conversiones

El *ADC* de este microcontrolador permite el inicio de secuencia de conversión/canal de dos formas:

1. Iniciadas por software: Las secuencias de conversión son iniciadas mediante código.
2. Iniciadas por hardware: Las secuencias de conversión son iniciadas dependiendo de otras señales, sean las mismas internas o externas al microcontrolador.

Calibración de hardware

Este periférico contiene un bloque de autocalibración, el cual debe ser utilizado luego de cada reinicio del microcontrolador o cada vez que se sale de modo de bajo consumo, para obtener la resolución y presición especificada por el fabricante.

La librería implementa la calibración por hardware en la función hal_adc_init

Velocidad de conversión

Cada conversión realizada toma un tiempo que dependerá del clock configurado en el periférico. Podemos obtener este tiempo de conversión mediante la ecuación: $t_{conv_{ADC}} = \frac{1}{25 * f_{ADC}}$

El multiplicador 25 en el denominador, es debido a la naturaleza del periférico de <e>aproximaciones sucesivas</e>. Esto implica que desde que se genera un inicio de conversión hasta que la misma finaliza, deben transcurrir 25 ciclos de clock del *ADC*.

Ejemplo: Configurando el *ADC* con una $f_{ADC} = 25MHz$ obtenemos el tiempo tomado por cada conversión:

$$t_{conv_{ADC}} = \frac{1}{25 * 1MHz}$$
$$t_{conv_{ADC}} = 1\mu s$$

Esto implica que entre un inicio de conversión y la finalización de la misma, pasará $1\mu s$. Nótese que este tiempo corresponde a una conversión para un único canal. En caso de estar convirtiendo varios canales, se deberá multiplicar $t_{conv_{ADC}}$ por la cantidad de canales activos en la secuencia de conversión, para obtener el tiempo total desde un inicio de secuencia de conversión y la finalización de todos los canales.

Ver también

Ejemplo_ADC.c

Estructuras de datos

- struct hal_adc_sequence_config_t
- struct hal_adc_sequence_result_t

Enumeraciones

- enum hal_adc_clock_source_en { HAL_ADC_CLOCK_SOURCE_FRO = 0, HAL_ADC_CLOCK_SYS_PLL }
- enum hal_adc_low_power_mode_en { HAL_ADC_LOW_POWER_MODE_DISABLED = 0, HAL_ADC_LOW_POWER_MODE_ENABLED }
- enum hal_adc_sequence_sel_en { HAL_ADC_SEQUENCE_SEL_A = 0, HAL_ADC_SEQUENCE_SEL_B }
- enum hal_adc_trigger_sel_en { HAL_ADC_TRIGGER_SEL_NONE = 0, HAL_ADC_TRIGGER_SEL_PININT0_IRQ, HAL_ADC_TRIGGER_SEL_PININT1_IRQ, HAL_ADC_TRIGGER_SEL_SCT0_OUT3, HAL_ADC_TRIGGER_SEL_SCT0_OUT4, HAL_ADC_TRIGGER_SEL_T0_MAT3, HAL_ADC_TRIGGER_SEL_CMP0_OUT_ADC, HAL_ADC_TRIGGER_SEL_GPIO_INT_BMAT, HAL_ADC_TRIGGER_SEL_ARM_TXEV }
- enum hal_adc_trigger_pol_sel_en { HAL_ADC_TRIGGER_POL_SEL_NEGATIVE_EDGE = 0, HAL_ADC_TRIGGER_POL_SEL_POSITIVE_EDGE }

- enum hal_adc_sync_sel_en { HAL_ADC_SYNC_SEL_ENABLE_SYNC = 0, HAL_ADC_SYNC_SEL_BYPASS_SYNC }
- enum hal_adc_interrupt_mode_en { HAL_ADC_INTERRUPT_MODE_EOC = 0, HAL_ADC_INTERRUPT_MODE_EOS }
- enum hal_adc_result_channel_en { HAL_ADC_RESULT_CHANNEL_0 = 0, HAL_ADC_RESULT_CHANNEL_1, HAL_ADC_RESULT_CHANNEL_2, HAL_ADC_RESULT_CHANNEL_3, HAL_ADC_RESULT_CHANNEL_4, HAL_ADC_RESULT_CHANNEL_5, HAL_ADC_RESULT_CHANNEL_6, HAL_ADC_RESULT_CHANNEL_7, HAL_ADC_RESULT_CHANNEL_8, HAL_ADC_RESULT_CHANNEL_9, HAL_ADC_RESULT_CHANNEL_10, HAL_ADC_RESULT_CHANNEL_11, HAL_ADC_RESULT_CHANNEL_GLOBAL }
- enum hal_adc_sequence_result_en { HAL_ADC_SEQUENCE_RESULT_VALID = 0, HAL_ADC_SEQUENCE_RESULT_INVALID }

Funciones

- void hal_adc_init_async_mode (uint32_t sample_freq, uint8_t div, hal_adc_clock_source_en clock_source, hal_adc_low_power_mode_en low_power)

  *Inicializar el ADC en modo **asincrónico**.*
- void hal_adc_init_sync_mode (uint32_t sample_freq, hal_adc_low_power_mode_en low_power)

  *Inicializar el ADC en modo **sincrónico**.*
- void hal_adc_deinit (void)

  De-inicialización del ADC.
- void hal_adc_config_sequence (hal_adc_sequence_sel_en sequence, const hal_adc_sequence_config_t *config)

  Configurar una secuencia de conversión.
- void hal_adc_enable_sequence (hal_adc_sequence_sel_en sequence)

  Habilitar una secuencia.
- void hal_adc_start_sequence (hal_adc_sequence_sel_en sequence)

  Disparar conversiones en una secuencia.
- hal_adc_sequence_result_en hal_adc_get_sequence_result (hal_adc_sequence_sel_en sequence, hal_adc_sequence_result_t *result)

  Obtener resultado de la secuencia.

5.1.2. Documentación de las estructuras de datos

5.1.2.1. struct hal_adc_sequence_result_t

Dato que representa el resultado de una conversión (sea de secuencia completa o de canal)

Ejemplos:

Ejemplo_ADC.c.

Campos de datos

hal_adc_result_channel_en	channel	Canal que generó el resultado
uint16_t	result	Valor de la conversión

5.1.3. Documentación de las enumeraciones

5.1.3.1. hal_adc_clock_source_en

`enum hal_adc_clock_source_en`

Selección de fuente de clock para el *ADC*

Valores de enumeraciones

HAL_ADC_CLOCK_SOURCE_FRO	Free running oscillator como fuente de clock
HAL_ADC_CLOCK_SYS_PLL	Phase locked loop oscillator como fuente de clock

5.1.3.2. hal_adc_low_power_mode_en

`enum hal_adc_low_power_mode_en`

Selección de modo bajo consumo

Valores de enumeraciones

HAL_ADC_LOW_POWER_MODE_DISABLED	Modo bajo consumo inhabilitado
HAL_ADC_LOW_POWER_MODE_ENABLED	Modo bajo consumo habilitado

5.1.3.3. hal_adc_sequence_sel_en

`enum hal_adc_sequence_sel_en`

Selección de secuencia de *ADC*

Valores de enumeraciones

HAL_ADC_SEQUENCE_SEL↩ _A	Secuencia A
HAL_ADC_SEQUENCE_SEL↩ _B	Secuencia B

5.1.3.4. hal_adc_trigger_sel_en

`enum hal_adc_trigger_sel_en`

Fuente de trigger para el *ADC*

Valores de enumeraciones

HAL_ADC_TRIGGER_SEL_NONE	Ninguna (trigger por software)
HAL_ADC_TRIGGER_SEL_PININT0_IRQ	Interrupción de PININT, canal 0
HAL_ADC_TRIGGER_SEL_PININT1_IRQ	Interrupción de PININT, canal 1
HAL_ADC_TRIGGER_SEL_SCT0_OUT3	Salida 3 del SCT
HAL_ADC_TRIGGER_SEL_SCT0_OUT4	Salida 4 del SCT
HAL_ADC_TRIGGER_SEL_T0_MAT3	Match 3 del CTIMER
HAL_ADC_TRIGGER_SEL_CMP0_OUT_ADC	Salida 0 del comparador analógico
HAL_ADC_TRIGGER_SEL_GPIO_INT_BMAT	Pattern match
HAL_ADC_TRIGGER_SEL_ARM_TXEV	Señal TXEV causada por una instrucción SEV

5.1.3.5. hal_adc_trigger_pol_sel_en

`enum hal_adc_trigger_pol_sel_en`

Selección de polaridad del trigger del *ADC*

Valores de enumeraciones

HAL_ADC_TRIGGER_POL_SEL_NEGATIVE_EDGE	Flanco negativo
HAL_ADC_TRIGGER_POL_SEL_POSITIVE_EDGE	Flanco positivo

5.1.3.6. hal_adc_sync_sel_en

`enum hal_adc_sync_sel_en`

Selección de sincronismo en secuencia del *ADC*

Valores de enumeraciones

HAL_ADC_SYNC_SEL_ENABLE_SYNC	Habilitación de sincronismo
HAL_ADC_SYNC_SEL_BYPASS_SYNC	Bypass el sincronismo

5.1.3.7. hal_adc_interrupt_mode_en

`enum hal_adc_interrupt_mode_en`

Selección de modo de interrupción del *ADC*

Valores de enumeraciones

HAL_ADC_INTERRUPT_MODE_EOC	Modo de interrupción en fin de conversión
HAL_ADC_INTERRUPT_MODE_EOS	Modo de interrupción en fin de secuencia

5.1.3.8. hal_adc_result_channel_en

`enum hal_adc_result_channel_en`

Canal que genero el resultado de *ADC*

Valores de enumeraciones

HAL_ADC_RESULT_CHANNEL_0	Canal 0
HAL_ADC_RESULT_CHANNEL_1	Canal 1
HAL_ADC_RESULT_CHANNEL_2	Canal 2
HAL_ADC_RESULT_CHANNEL_3	Canal 3
HAL_ADC_RESULT_CHANNEL_4	Canal 4
HAL_ADC_RESULT_CHANNEL_5	Canal 5
HAL_ADC_RESULT_CHANNEL_6	Canal 6
HAL_ADC_RESULT_CHANNEL_7	Canal 7
HAL_ADC_RESULT_CHANNEL_8	Canal 8
HAL_ADC_RESULT_CHANNEL_9	Canal 9
HAL_ADC_RESULT_CHANNEL_10	Canal 10
HAL_ADC_RESULT_CHANNEL_11	Canal 11
HAL_ADC_RESULT_CHANNEL_GLOBAL	Global

5.1.3.9. hal_adc_sequence_result_en

`enum hal_adc_sequence_result_en`

Resultado de obtención de resultado de secuencia

Valores de enumeraciones

HAL_ADC_SEQUENCE_RESULT_VALID	Resultado válido
HAL_ADC_SEQUENCE_RESULT_INVALID	Resultado inválido

5.1.4. Documentación de las funciones

5.1.4.1. hal_adc_init_async_mode()

```
void hal_adc_init_async_mode (
            uint32_t sample_freq,
            uint8_t div,
            hal_adc_clock_source_en clock_source,
            hal_adc_low_power_mode_en low_power )
```

Inicializar el *ADC* en modo **asincrónico**.

Realiza la calibración de hardware y fija la frecuencia de muestreo deseada. Nota: Solamente se debe realizar el llamado a una de las dos funciones de inicialización del *ADC*

Ver también

hal_adc_clock_source_en
hal_adc_low_power_mode_en

Parámetros

in	*sample_freq*	Frecuencia de sampleo deseada
in	*div*	Divisor para la lógica del *ADC*
in	*clock_source*	Fuente de clock para el *ADC*
in	*low_power*	Selección de modo de bajo consumo

5.1.4.2. hal_adc_init_sync_mode()

```
void hal_adc_init_sync_mode (
            uint32_t sample_freq,
            hal_adc_low_power_mode_en low_power )
```

Inicializar el *ADC* en modo **sincrónico**.

Realiza la calibración de hardware y fija la frecuencia de muestreo deseada.

Ver también

hal_adc_clock_source_en
hal_adc_operation_mode_en
hal_adc_low_power_mode_en

Parámetros

in	*sample_freq*	Frecuencia de sampleo deseada
in	*low_power*	Selección de modo de bajo consumo

Ejemplos:

Ejemplo_ADC.c.

5.1.4.3. hal_adc_config_sequence()

```
void hal_adc_config_sequence (
            hal_adc_sequence_sel_en sequence,
            const hal_adc_sequence_config_t * config )
```

Configurar una secuencia de conversión.

Esta función no habilita la secuencia, al menos que el parametro **burst** este activo

Ver también

hal_adc_sequence_sel_en
hal_adc_sequence_config_t

Parámetros

in	*sequence*	Seleccion de secuencia a configurar
in	*config*	Configuracion deseada para la secuencia

Ejemplos:

Ejemplo_ADC.c.

5.1.4.4. hal_adc_enable_sequence()

```
void hal_adc_enable_sequence (
            hal_adc_sequence_sel_en sequence )
```

Habilitar una secuencia.

Ver también

hal_adc_sequence_sel_en

Parámetros

in	*sequence*	Secuencia a habilitar

5.1.4.5. hal_adc_start_sequence()

```
void hal_adc_start_sequence (
            hal_adc_sequence_sel_en sequence )
```

Disparar conversiones en una secuencia.

La configuración de la secuencia, en particular el parametro **single_step**, influye en si esta funcion dispara una secuencia entera o un paso de la misma.

Ver también

hal_adc_sequence_sel_en

Parámetros

in	*sequence*	Secuencia a disparar

Ejemplos:

Ejemplo_ADC.c.

5.1.4.6. hal_adc_get_sequence_result()

```
hal_adc_sequence_result_en hal_adc_get_sequence_result (
            hal_adc_sequence_sel_en sequence,
            hal_adc_sequence_result_t * result )
```

Obtener resultado de la secuencia.

El comportamiento de esta funcion depende de la configuración de la secuencia, en particular de la configuracion **MODE**. En caso de estar configurada para interrumpir al final de cada conversión, la función únicamente guardara el resultado de la conversión en el primer lugar del parámetro <e>result</e>, caso contrario, guardara la cantidad de canales habilitados en la conversión en los distintos lugares del parámetro <e>result</e>.

Ver también

hal_adc_sequence_result_en
hal_adc_sequence_sel_en
hal_adc_sequence_result_t

Parámetros

in	*sequence*	Secuencia de la cual obtener el resultado
out	*result*	Lugares donde guardar los resultados de la secuencia

Devuelve

Resultado de la función

Ejemplos:

Ejemplo_ADC.c.

Capítulo 6

Documentación de las estructuras de datos

6.1. Referencia de la Estructura CTIMER_CC_config_t

Campos de datos

- uint8_t **rising_edge_capture**: 1
- uint8_t **falling_edge_capture**: 1
- uint8_t **interrupt_on_capture**: 1
- void(∗ **capture_callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hpl/HPL_CTIMER.h

6.2. Referencia de la Estructura CTIMER_MR_config_t

Campos de datos

- uint8_t **interrupt_on_match**: 1
- uint8_t **reset_on_match**: 1
- uint8_t **stop_on_match**: 1
- uint8_t **reload_on_match**: 1
- uint32_t **match_value**
- void(∗ **match_callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hpl/HPL_CTIMER.h

6.3. Referencia de la Estructura hal_adc_sequence_config_t

```
#include <HAL_ADC.h>
```

Campos de datos

- uint16_t channels
- hal_adc_trigger_sel_en trigger
- hal_adc_trigger_pol_sel_en trigger_pol
- hal_adc_sync_sel_en sync_bypass
- hal_adc_interrupt_mode_en mode
- uint8_t **burst**
- uint8_t single_step
- uint8_t low_priority
- void(∗ callback)(void)

6.3.1. Descripción detallada

Configuración de secuencia de *ADC*

Ejemplos:

Ejemplo_ADC.c.

6.3.2. Documentación de los campos

6.3.2.1. channels

```
uint16_t hal_adc_sequence_config_t::channels
```

Canales habilitados. Cada uno de los bits representa el canal

Ejemplos:

Ejemplo_ADC.c.

6.3.2.2. trigger

```
hal_adc_trigger_sel_en hal_adc_sequence_config_t::trigger
```

Configuración de fuente de trigger para la secuencia

6.3.2.3. trigger_pol

```
hal_adc_trigger_pol_sel_en hal_adc_sequence_config_t::trigger_pol
```

Configuración de flanco del trigger para la secuencia

6.3.2.4. sync_bypass

`hal_adc_sync_sel_en hal_adc_sequence_config_t::sync_bypass`

Configuración de sincronismo de la secuencia

6.3.2.5. mode

`hal_adc_interrupt_mode_en hal_adc_sequence_config_t::mode`

Configuración de modo de interrupcion

6.3.2.6. single_step

`uint8_t hal_adc_sequence_config_t::single_step`

Configuración de modo BURST. En caso de ser 0 esta inhabilitado, cualquier otro valor lo habilita Configuración de funcionamiento del trigger. En caso de ser 0, un trigger dispara la conversión de toda la secuencia configurada, en caso de ser cualquier otro valor, un trigger dispara la conversión del siguiente canal habilitado en la secuencia

6.3.2.7. low_priority

`uint8_t hal_adc_sequence_config_t::low_priority`

Fijar baja prioridad de la secuencia. Unicamente aplica para la secuencia *A*. En caso de ser 0, la secuencia *A* tiene prioridad por sobre el *B*, cualquier otro valor, implica que la secuencia B tiene prioridad por sobre la *A*

6.3.2.8. callback

`void(* hal_adc_sequence_config_t::callback) (void)`

Callback a ejecutar en interrupción de secuencia. La misma se generará al final de la conversión de cada canal, o de toda la secuencia, dependiendo de la configuración global del *ADC*

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_ADC.h

6.4. Referencia de la Estructura hal_ctimer_match_config_t

Campos de datos

- uint8_t **interrupt_on_match**
- uint8_t **reset_on_match**
- uint8_t **stop_on_match**
- uint8_t **reload_on_match**
- uint32_t **match_value_useg**
- hal_ctimer_match_action_en **match_action**
- uint8_t **enable_external_pin**
- hal_gpio_portpin_en **match_pin**
- void(* **callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_CTIMER.h

6.5. Referencia de la Estructura hal_ctimer_pwm_channel_config_t

Campos de datos

- uint8_t **interrupt_on_action**
- uint32_t duty

  Duty en decimas de porciento (1 equivale a 0.1 %)
- hal_gpio_portpin_en **channel_pin**
- void(∗ **callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_CTIMER.h

6.6. Referencia de la Estructura hal_ctimer_pwm_config_t

Campos de datos

- uint32_t clock_div

  Corresponde al numero deseado a dividir menos 1.
- uint32_t pwm_period_useg

  Periodo del PWM en microsegundos.
- uint8_t **interrupt_on_period**
- void(∗ **callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_CTIMER.h

6.7. Referencia de la Estructura hal_pinint_config_t

Campos de datos

- hal_pinint_channel_en **channel**
- hal_pinint_interrupt_mode_en **mode**
- hal_pinint_level_int_en **int_on_level**
- uint8_t **int_on_rising_edge**
- uint8_t **int_on_falling_edge**
- hal_gpio_portpin_en **portpin**
- void(∗ **callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_PININT.h

6.8. Referencia de la Estructura hal_spi_master_mode_config_t

Campos de datos

- hal_syscon_peripheral_clock_sel_en **clock_source**
- uint8_t **pre_delay**
- uint8_t **post_delay**
- uint8_t **frame_delay**
- uint8_t **transfer_delay**
- hal_gpio_portpin_en **sck_portpin**
- hal_gpio_portpin_en **miso_portpin**
- hal_gpio_portpin_en **mosi_portpin**
- hal_gpio_portpin_en **ssel_portpin** [4]
- hal_spi_ssel_polarity_en **ssel_polarity** [4]
- void(∗ **tx_free_callback**)(void)
- void(∗ **rx_ready_callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_SPI.h

6.9. Referencia de la Estructura hal_uart_config_t

Campos de datos

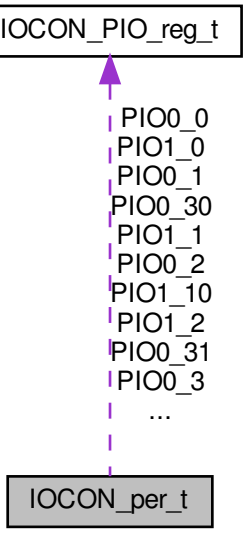
- hal_uart_datalen_en **data_length**
- hal_uart_parity_en **parity**
- hal_uart_stop_en **stop_bits**
- hal_uart_oversampling_en **oversampling**
- hal_syscon_peripheral_clock_sel_en **clock_selection**
- uint32_t **baudrate**
- hal_gpio_portpin_en **tx_portpin**
- hal_gpio_portpin_en **rx_portpin**
- void(∗ **rx_ready_callback**)(void)
- void(∗ **tx_ready_callback**)(void)

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hal/HAL_UART.h

6.10. Referencia de la Estructura IOCON_per_t

Diagrama de colaboración para IOCON_per_t:



Campos de datos

- IOCON_PIO_reg_t **PIO0_17**
- IOCON_PIO_reg_t **PIO0_13**
- IOCON_PIO_reg_t **PIO0_12**
- IOCON_PIO_reg_t **PIO0_5**
- IOCON_PIO_reg_t **PIO0_4**
- IOCON_PIO_reg_t **PIO0_3**
- IOCON_PIO_reg_t **PIO0_2**
- IOCON_PIO_reg_t **PIO0_11**
- IOCON_PIO_reg_t **PIO0_10**
- IOCON_PIO_reg_t **PIO0_16**
- IOCON_PIO_reg_t **PIO0_15**
- IOCON_PIO_reg_t **PIO0_1**
- const uint32_t **_RESERVED_1**
- IOCON_PIO_reg_t **PIO0_9**
- IOCON_PIO_reg_t **PIO0_8**
- IOCON_PIO_reg_t **PIO0_7**
- IOCON_PIO_reg_t **PIO0_6**
- IOCON_PIO_reg_t **PIO0_0**
- IOCON_PIO_reg_t **PIO0_14**
- const uint32_t **_RESERVED_2**
- IOCON_PIO_reg_t **PIO0_28**
- IOCON_PIO_reg_t **PIO0_27**
- IOCON_PIO_reg_t **PIO0_26**

- IOCON_PIO_reg_t **PIO0_25**
- IOCON_PIO_reg_t **PIO0_24**
- IOCON_PIO_reg_t **PIO0_23**
- IOCON_PIO_reg_t **PIO0_22**
- IOCON_PIO_reg_t **PIO0_21**
- IOCON_PIO_reg_t **PIO0_20**
- IOCON_PIO_reg_t **PIO0_19**
- IOCON_PIO_reg_t **PIO0_18**
- IOCON_PIO_reg_t **PIO1_8**
- IOCON_PIO_reg_t **PIO1_9**
- IOCON_PIO_reg_t **PIO1_12**
- IOCON_PIO_reg_t **PIO1_13**
- IOCON_PIO_reg_t **PIO0_31**
- IOCON_PIO_reg_t **PIO1_0**
- IOCON_PIO_reg_t **PIO1_1**
- IOCON_PIO_reg_t **PIO1_2**
- IOCON_PIO_reg_t **PIO1_14**
- IOCON_PIO_reg_t **PIO1_15**
- IOCON_PIO_reg_t **PIO1_3**
- IOCON_PIO_reg_t **PIO1_4**
- IOCON_PIO_reg_t **PIO1_5**
- IOCON_PIO_reg_t **PIO1_16**
- IOCON_PIO_reg_t **PIO1_17**
- IOCON_PIO_reg_t **PIO1_6**
- IOCON_PIO_reg_t **PIO1_18**
- IOCON_PIO_reg_t **PIO1_19**
- IOCON_PIO_reg_t **PIO1_7**
- IOCON_PIO_reg_t **PIO0_29**
- IOCON_PIO_reg_t **PIO0_30**
- IOCON_PIO_reg_t **PIO1_20**
- IOCON_PIO_reg_t **PIO1_21**
- IOCON_PIO_reg_t **PIO1_11**
- IOCON_PIO_reg_t **PIO1_10**

6.10.1. Documentación de los campos

6.10.1.1. PIO0_17

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_17
```

6.10.1.2. PIO0_13

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_13
```

6.10.1.3. PIO0_12

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_12
```

6.10.1.4. PIO0_5

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_5
```

6.10.1.5. PIO0_4

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_4
```

6.10.1.6. PIO0_3

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_3
```

6.10.1.7. PIO0_2

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_2
```

6.10.1.8. PIO0_11

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_11
```

6.10.1.9. PIO0_10

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_10
```

6.10.1.10. PIO0_16

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_16
```

6.10.1.11. PIO0_15

IOCON_PIO_reg_t IOCON_per_t::PIO0_15

6.10.1.12. PIO0_1

IOCON_PIO_reg_t IOCON_per_t::PIO0_1

6.10.1.13. _RESERVED_1

const uint32_t IOCON_per_t::_RESERVED_1

6.10.1.14. PIO0_9

IOCON_PIO_reg_t IOCON_per_t::PIO0_9

6.10.1.15. PIO0_8

IOCON_PIO_reg_t IOCON_per_t::PIO0_8

6.10.1.16. PIO0_7

IOCON_PIO_reg_t IOCON_per_t::PIO0_7

6.10.1.17. PIO0_6

IOCON_PIO_reg_t IOCON_per_t::PIO0_6

6.10.1.18. PIO0_0

IOCON_PIO_reg_t IOCON_per_t::PIO0_0

6.10.1.19. PIO0_14

IOCON_PIO_reg_t IOCON_per_t::PIO0_14

6.10.1.20. _RESERVED_2

const uint32_t IOCON_per_t::_RESERVED_2

6.10.1.21. PIO0_28

IOCON_PIO_reg_t IOCON_per_t::PIO0_28

6.10.1.22. PIO0_27

IOCON_PIO_reg_t IOCON_per_t::PIO0_27

6.10.1.23. PIO0_26

IOCON_PIO_reg_t IOCON_per_t::PIO0_26

6.10.1.24. PIO0_25

IOCON_PIO_reg_t IOCON_per_t::PIO0_25

6.10.1.25. PIO0_24

IOCON_PIO_reg_t IOCON_per_t::PIO0_24

6.10.1.26. PIO0_23

IOCON_PIO_reg_t IOCON_per_t::PIO0_23

6.10.1.27. PIO0_22

IOCON_PIO_reg_t IOCON_per_t::PIO0_22

6.10.1.28. PIO0_21

IOCON_PIO_reg_t IOCON_per_t::PIO0_21

6.10.1.29. PIO0_20

IOCON_PIO_reg_t IOCON_per_t::PIO0_20

6.10.1.30. PIO0_19

IOCON_PIO_reg_t IOCON_per_t::PIO0_19

6.10.1.31. PIO0_18

IOCON_PIO_reg_t IOCON_per_t::PIO0_18

6.10.1.32. PIO1_8

IOCON_PIO_reg_t IOCON_per_t::PIO1_8

6.10.1.33. PIO1_9

IOCON_PIO_reg_t IOCON_per_t::PIO1_9

6.10.1.34. PIO1_12

IOCON_PIO_reg_t IOCON_per_t::PIO1_12

6.10.1.35. PIO1_13

IOCON_PIO_reg_t IOCON_per_t::PIO1_13

6.10.1.36. PIO0_31

IOCON_PIO_reg_t IOCON_per_t::PIO0_31

6.10.1.37. PIO1_0

IOCON_PIO_reg_t IOCON_per_t::PIO1_0

6.10.1.38. PIO1_1

IOCON_PIO_reg_t IOCON_per_t::PIO1_1

6.10.1.39. PIO1_2

IOCON_PIO_reg_t IOCON_per_t::PIO1_2

6.10.1.40. PIO1_14

IOCON_PIO_reg_t IOCON_per_t::PIO1_14

6.10.1.41. PIO1_15

IOCON_PIO_reg_t IOCON_per_t::PIO1_15

6.10.1.42. PIO1_3

IOCON_PIO_reg_t IOCON_per_t::PIO1_3

6.10.1.43. PIO1_4

IOCON_PIO_reg_t IOCON_per_t::PIO1_4

6.10.1.44. PIO1_5

IOCON_PIO_reg_t IOCON_per_t::PIO1_5

6.10.1.45. PIO1_16

IOCON_PIO_reg_t IOCON_per_t::PIO1_16

6.10.1.46. PIO1_17

IOCON_PIO_reg_t IOCON_per_t::PIO1_17

6.10.1.47. PIO1_6

IOCON_PIO_reg_t IOCON_per_t::PIO1_6

6.10.1.48. PIO1_18

IOCON_PIO_reg_t IOCON_per_t::PIO1_18

6.10.1.49. PIO1_19

IOCON_PIO_reg_t IOCON_per_t::PIO1_19

6.10.1.50. PIO1_7

IOCON_PIO_reg_t IOCON_per_t::PIO1_7

6.10.1.51. PIO0_29

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_29
```

6.10.1.52. PIO0_30

```
IOCON_PIO_reg_t IOCON_per_t::PIO0_30
```

6.10.1.53. PIO1_20

```
IOCON_PIO_reg_t IOCON_per_t::PIO1_20
```

6.10.1.54. PIO1_21

```
IOCON_PIO_reg_t IOCON_per_t::PIO1_21
```

6.10.1.55. PIO1_11

```
IOCON_PIO_reg_t IOCON_per_t::PIO1_11
```

6.10.1.56. PIO1_10

```
IOCON_PIO_reg_t IOCON_per_t::PIO1_10
```

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hri/HRI_IOCON.h

6.11. Referencia de la Estructura IOCON_PIO_reg_t

Campos de datos

- uint32_t **__pad0__**: 3
- uint32_t **MODE**: 2
- uint32_t **HYS**: 1
- uint32_t **INV**: 1
- uint32_t **I2CMODE**: 2
- uint32_t **__pad1__**: 1
- uint32_t **OD**: 1
- uint32_t **S_MODE**: 2
- uint32_t **CLK_DIV**: 3
- uint32_t **DACMODE**: 1
- uint32_t **__pad2__**: 15

6.11.1. Documentación de los campos

6.11.1.1. __pad0__

```
uint32_t IOCON_PIO_reg_t::__pad0__
```

6.11.1.2. MODE

```
uint32_t IOCON_PIO_reg_t::MODE
```

6.11.1.3. HYS

```
uint32_t IOCON_PIO_reg_t::HYS
```

6.11.1.4. INV

```
uint32_t IOCON_PIO_reg_t::INV
```

6.11.1.5. I2CMODE

```
uint32_t IOCON_PIO_reg_t::I2CMODE
```

6.11.1.6. __pad1__

```
uint32_t IOCON_PIO_reg_t::__pad1__
```

6.11.1.7. OD

```
uint32_t IOCON_PIO_reg_t::OD
```

6.11.1.8. S_MODE

```
uint32_t IOCON_PIO_reg_t::S_MODE
```

6.11.1.9. CLK_DIV

```
uint32_t IOCON_PIO_reg_t::CLK_DIV
```

6.11.1.10. DACMODE

```
uint32_t IOCON_PIO_reg_t::DACMODE
```

6.11.1.11. __pad2__

```
uint32_t IOCON_PIO_reg_t::__pad2__
```

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/hri/HRI_IOCON.h

Capítulo 7

Documentación de archivos

7.1. Referencia del Archivo includes/hal/HAL_ADC.h

Declaraciones a nivel de aplicacion del periferico ADC (LPC845)

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

Dependencia gráfica adjunta para HAL_ADC.h:

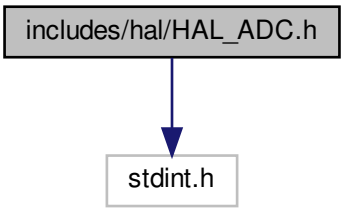
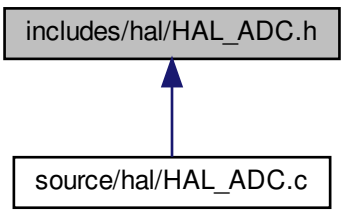


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct hal_adc_sequence_config_t
- struct hal_adc_sequence_result_t

Enumeraciones

- enum hal_adc_clock_source_en { HAL_ADC_CLOCK_SOURCE_FRO = 0, HAL_ADC_CLOCK_SYS_PLL }
- enum hal_adc_low_power_mode_en { HAL_ADC_LOW_POWER_MODE_DISABLED = 0, HAL_ADC_LOW_POWER_MODE_ENABLED }
- enum hal_adc_sequence_sel_en { HAL_ADC_SEQUENCE_SEL_A = 0, HAL_ADC_SEQUENCE_SEL_B }
- enum hal_adc_trigger_sel_en { HAL_ADC_TRIGGER_SEL_NONE = 0, HAL_ADC_TRIGGER_SEL_PININT0_IRQ, HAL_ADC_TRIGGER_SEL_PININT1_IRQ, HAL_ADC_TRIGGER_SEL_SCT0_OUT3, HAL_ADC_TRIGGER_SEL_SCT0_OUT4, HAL_ADC_TRIGGER_SEL_T0_MAT3, HAL_ADC_TRIGGER_SEL_CMP0_OUT_ADC, HAL_ADC_TRIGGER_SEL_GPIO_INT_BMAT, HAL_ADC_TRIGGER_SEL_ARM_TXEV }
- enum hal_adc_trigger_pol_sel_en { HAL_ADC_TRIGGER_POL_SEL_NEGATIVE_EDGE = 0, HAL_ADC_TRIGGER_POL_SEL_POSITIVE_EDGE }
- enum hal_adc_sync_sel_en { HAL_ADC_SYNC_SEL_ENABLE_SYNC = 0, HAL_ADC_SYNC_SEL_BYPASS_SYNC }
- enum hal_adc_interrupt_mode_en { HAL_ADC_INTERRUPT_MODE_EOC = 0, HAL_ADC_INTERRUPT_MODE_EOS }
- enum hal_adc_result_channel_en { HAL_ADC_RESULT_CHANNEL_0 = 0, HAL_ADC_RESULT_CHANNEL_1, HAL_ADC_RESULT_CHANNEL_2, HAL_ADC_RESULT_CHANNEL_3, HAL_ADC_RESULT_CHANNEL_4, HAL_ADC_RESULT_CHANNEL_5, HAL_ADC_RESULT_CHANNEL_6, HAL_ADC_RESULT_CHANNEL_7, HAL_ADC_RESULT_CHANNEL_8, HAL_ADC_RESULT_CHANNEL_9, HAL_ADC_RESULT_CHANNEL_10, HAL_ADC_RESULT_CHANNEL_11, HAL_ADC_RESULT_CHANNEL_GLOBAL }
- enum hal_adc_sequence_result_en { HAL_ADC_SEQUENCE_RESULT_VALID = 0, HAL_ADC_SEQUENCE_RESULT_INVALID }

Funciones

- void hal_adc_init_async_mode (uint32_t sample_freq, uint8_t div, hal_adc_clock_source_en clock_source, hal_adc_low_power_mode_en low_power)

  *Inicializar el ADC en modo **asincrónico**.*
- void hal_adc_init_sync_mode (uint32_t sample_freq, hal_adc_low_power_mode_en low_power)

  *Inicializar el ADC en modo **sincrónico**.*
- void hal_adc_deinit (void)

  De-inicialización del ADC.
- void hal_adc_config_sequence (hal_adc_sequence_sel_en sequence, const hal_adc_sequence_config_t *config)

  Configurar una secuencia de conversión.
- void hal_adc_enable_sequence (hal_adc_sequence_sel_en sequence)

  Habilitar una secuencia.
- void hal_adc_start_sequence (hal_adc_sequence_sel_en sequence)

  Disparar conversiones en una secuencia.
- hal_adc_sequence_result_en hal_adc_get_sequence_result (hal_adc_sequence_sel_en sequence, hal_adc_sequence_result_t *result)

  Obtener resultado de la secuencia.

7.1.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico ADC (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.2. Referencia del Archivo includes/hal/HAL_CTIMER.h

Declaraciones a nivel de aplicacion del periferico CTIMER (LPC845)

```
#include <stdint.h>
#include <HAL_GPIO.h>
```

Dependencia gráfica adjunta para HAL_CTIMER.h:

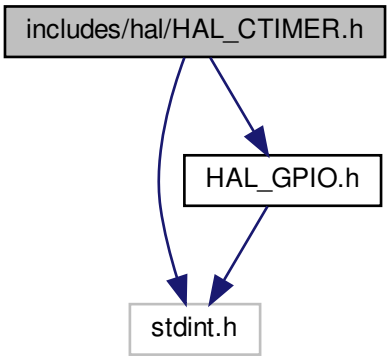
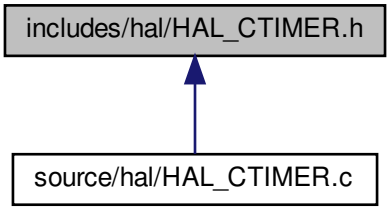


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct hal_ctimer_match_config_t
- struct hal_ctimer_pwm_channel_config_t
- struct hal_ctimer_pwm_config_t

Enumeraciones

- enum **hal_ctimer_match_action_en** { **HAL_CTIMER_MATCH_DO_NOGHING** = 0, **HAL_CTIMER_MATCH_CLEAR_PIN**, **HAL_CTIMER_MATCH_SET_PIN**, **HAL_CTIMER_MATCH_TOGGLE_PIN** }
- enum **hal_ctimer_match_sel_en** { **HAL_CTIMER_MATCH_0** = 0, **HAL_CTIMER_MATCH_1**, **HAL_CTIMER_MATCH_2**, **HAL_CTIMER_MATCH_3** }
- enum **hal_ctimer_pwm_channel_sel_en** { **HAL_CTIMER_PWM_CHANNEL_0** = 0, **HAL_CTIMER_PWM_CHANNEL_1**, **HAL_CTIMER_PWM_CHANNEL_2** }

Funciones

- void hal_ctimer_timer_mode_init (uint32_t clock_div)

  Inicializacion del periferico en modo timer.
- void hal_ctimer_timer_mode_config_match (hal_ctimer_match_sel_en match_sel, const hal_ctimer_match_config_t *match_config)

  Configurar un canal de match.
- void hal_ctimer_timer_mode_run (void)

  Habilitar el conteo del ctimer.
- void hal_ctimer_timer_mode_stop (void)

  Inhabilitar el conteo del ctimer.
- void hal_ctimer_timer_mode_reset (void)

  Reiniciar el conteo del ctimer.
- void hal_ctimer_pwm_mode_init (const hal_ctimer_pwm_config_t *config)

  Inicializar el CTIMER en modo PWM.
- void hal_ctimer_pwm_mode_set_period (uint32_t period_useg)

  Actualizar el periodo en modo PWM.
- void hal_ctimer_pwm_mode_config_channel (hal_ctimer_pwm_channel_sel_en channel_sel, const hal_ctimer_pwm_channel_config_t *channel_config)

  Actualizar configuracion de algun canal de PWM.

7.2.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico CTIMER (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.2.2. Documentación de las funciones

7.2.2.1. hal_ctimer_timer_mode_init()

```
void hal_ctimer_timer_mode_init (
            uint32_t clock_div )
```

Inicializacion del periferico en modo timer.

Esta funcion no pone a correr el contador.

Parámetros

in	*clock_div*	Divisor del clock principal deseado (el valor efectivo es este valor + 1)

7.2.2.2. hal_ctimer_timer_mode_config_match()

```
void hal_ctimer_timer_mode_config_match (
            hal_ctimer_match_sel_en match_sel,
            const hal_ctimer_match_config_t * match_config )
```

Configurar un canal de match.

Parámetros

in	*match_sel*	Match a configurar
in	*match_config*	Configuracion deseada

7.2.2.3. hal_ctimer_pwm_mode_init()

```
void hal_ctimer_pwm_mode_init (
            const hal_ctimer_pwm_config_t * config )
```

Inicializar el CTIMER en modo PWM.

Parámetros

in	*config*	Configuracion deseada

7.2.2.4. hal_ctimer_pwm_mode_set_period()

```
void hal_ctimer_pwm_mode_set_period (
            uint32_t period_useg )
```

Actualizar el periodo en modo PWM.

Parámetros

in	*period_useg*	Nuevo periodo deseado en microsegundos

7.2.2.5. hal_ctimer_pwm_mode_config_channel()

```
void hal_ctimer_pwm_mode_config_channel (
            hal_ctimer_pwm_channel_sel_en channel_sel,
            const hal_ctimer_pwm_channel_config_t * channel_config )
```

Actualizar configuracion de algun canal de PWM.

Parámetros

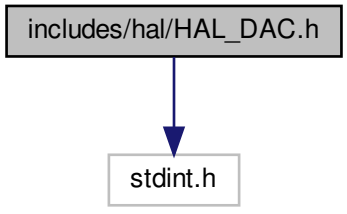
in	*channel_sel*	Seleccion de canal a configurar
in	*channel_config*	Configuracion del canal de PWM

7.3. Referencia del Archivo includes/hal/HAL_DAC.h

Declaraciones a nivel de aplicacion del periferico DAC (LPC845)

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

Dependencia gráfica adjunta para HAL_DAC.h:



Estructuras de datos

- struct hal_dac_ctrl_config_t

Enumeraciones

- enum **hal_dac_en** { **HAL_DAC_0** = 0, **HAL_DAC_1** }
- enum **hal_dac_settling_time_en** { **HAL_DAC_SETTLING_TIME_1US_MAX** = 0, **HAL_DAC_SETTLIN↩ G_TIME_2_5US_MAX** }

Funciones

- void hal_dac_init (hal_dac_en dac, hal_dac_settling_time_en settling_time, uint32_t initial_value)

  Inicializacion del DAC.

7.3.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico DAC (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.3.2. Documentación de las estructuras de datos

7.3.2.1. struct hal_dac_ctrl_config_t

Campos de datos

uint8_t	count_enable: 1	
uint8_t	double_buffering: 1	
uint8_t	dma_enable: 1	
uint8_t	dma_request: 1	

7.3.3. Documentación de las funciones

7.3.3.1. hal_dac_init()

```
void hal_dac_init (
            hal_dac_en dac,
```

```
            hal_dac_settling_time_en settling_time,
            uint32_t initial_value )
```

Inicializacion del DAC.

Parámetros

in	*dac*	Cual de los dos DACs inicializar
in	*settling_time*	Velocidad de conversion del DAC
in	*initial_value*	Valor inicial del DAC

7.4. Referencia del Archivo includes/hal/HAL_GPIO.h

Declaraciones a nivel de aplicacion del periferico GPIO (LPC845)

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

Dependencia gráfica adjunta para HAL_GPIO.h:

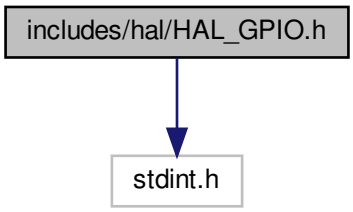
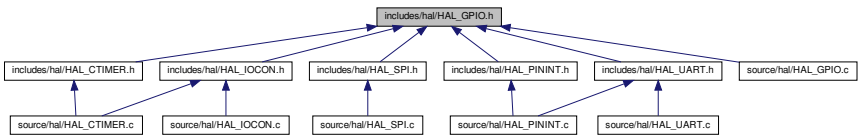


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



defines

- #define **HAL_GPIO_PORTPIN_TO_PORT**(x) (x / 32)
- #define **HAL_GPIO_PORTPIN_TO_PIN**(x) (x % 32)

Enumeraciones

- enum **hal_gpio_port_en** { **HAL_GPIO_PORT_0** = 0, **HAL_GPIO_PORT_1** }
- enum **hal_gpio_portpin_en** {
  HAL_GPIO_PORTPIN_0_0 = 0, **HAL_GPIO_PORTPIN_0_1**, **HAL_GPIO_PORTPIN_0_2**, **HAL_GPIO_P↩ ORTPIN_0_3**,
  HAL_GPIO_PORTPIN_0_4, **HAL_GPIO_PORTPIN_0_5**, **HAL_GPIO_PORTPIN_0_6**, **HAL_GPIO_POR↩ TPIN_0_7**,
  HAL_GPIO_PORTPIN_0_8, **HAL_GPIO_PORTPIN_0_9**, **HAL_GPIO_PORTPIN_0_10**, **HAL_GPIO_PO↩ RTPIN_0_11**,
  HAL_GPIO_PORTPIN_0_12, **HAL_GPIO_PORTPIN_0_13**, **HAL_GPIO_PORTPIN_0_14**, **HAL_GPIO_P↩ ORTPIN_0_15**,
  HAL_GPIO_PORTPIN_0_16, **HAL_GPIO_PORTPIN_0_17**, **HAL_GPIO_PORTPIN_0_18**, **HAL_GPIO_P↩ ORTPIN_0_19**,
  HAL_GPIO_PORTPIN_0_20, **HAL_GPIO_PORTPIN_0_21**, **HAL_GPIO_PORTPIN_0_22**, **HAL_GPIO_P↩ ORTPIN_0_23**,
  HAL_GPIO_PORTPIN_0_24, **HAL_GPIO_PORTPIN_0_25**, **HAL_GPIO_PORTPIN_0_26**, **HAL_GPIO_P↩ ORTPIN_0_27**,
  HAL_GPIO_PORTPIN_0_28, **HAL_GPIO_PORTPIN_0_29**, **HAL_GPIO_PORTPIN_0_30**, **HAL_GPIO_P↩ ORTPIN_0_31**,
  HAL_GPIO_PORTPIN_1_0, **HAL_GPIO_PORTPIN_1_1**, **HAL_GPIO_PORTPIN_1_2**, **HAL_GPIO_POR↩ TPIN_1_3**,
  HAL_GPIO_PORTPIN_1_4, **HAL_GPIO_PORTPIN_1_5**, **HAL_GPIO_PORTPIN_1_6**, **HAL_GPIO_POR↩ TPIN_1_7**,
  HAL_GPIO_PORTPIN_1_8, **HAL_GPIO_PORTPIN_1_9**, **HAL_GPIO_PORTPIN_NOT_USED** }
- enum **hal_gpio_dir_en** { **HAL_GPIO_DIR_INPUT** = 0, **HAL_GPIO_DIR_OUTPUT** }

Funciones

- void hal_gpio_init (hal_gpio_port_en port)

  Inicializar un puerto.
- void hal_gpio_set_dir (hal_gpio_portpin_en portpin, hal_gpio_dir_en dir, uint8_t initial_state)

  Fijar direccion de una GPIO.
- void hal_gpio_set_pin (hal_gpio_portpin_en portpin)

  Fijar estado activo de una GPIO.
- void hal_gpio_clear_pin (hal_gpio_portpin_en portpin)

  Fijar estado inactivo de una GPIO.
- void hal_gpio_toggle_pin (hal_gpio_portpin_en portpin)

  Invertir estado de una GPIO.
- uint8_t hal_gpio_read_pin (hal_gpio_portpin_en portpin)

  Leer el estado de una GPIO.

7.4.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico GPIO (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.4.2. Documentación de las funciones

7.4.2.1. hal_gpio_init()

```
void hal_gpio_init (
            hal_gpio_port_en port )
```

Inicializar un puerto.

Parámetros

in	*port*	Puerto a inicializar

7.4.2.2. hal_gpio_set_dir()

```
void hal_gpio_set_dir (
            hal_gpio_portpin_en portpin,
            hal_gpio_dir_en dir,
            uint8_t initial_state )
```

Fijar direccion de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a configurar
in	*dir*	Direccion deseada
in	*initial_state*	Estado inicial (aplica para salidas nada mas)

7.4.2.3. hal_gpio_set_pin()

```
void hal_gpio_set_pin (
            hal_gpio_portpin_en portpin )
```

Fijar estado activo de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

7.4.2.4. hal_gpio_clear_pin()

```
void hal_gpio_clear_pin (
            hal_gpio_portpin_en portpin )
```

Fijar estado inactivo de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

7.4.2.5. hal_gpio_toggle_pin()

```
void hal_gpio_toggle_pin (
            hal_gpio_portpin_en portpin )
```

Invertir estado de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

7.4.2.6. hal_gpio_read_pin()

```
uint8_t hal_gpio_read_pin (
            hal_gpio_portpin_en portpin )
```

Leer el estado de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

Devuelve

Estado actual de la GPIO

7.5. Referencia del Archivo includes/hal/HAL_IOCON.h

Declaraciones a nivel de aplicacion del periferico IOCON (LPC845)

```
#include <HPL_IOCON.h>
#include <HAL_GPIO.h>
```

Dependencia gráfica adjunta para HAL_IOCON.h:

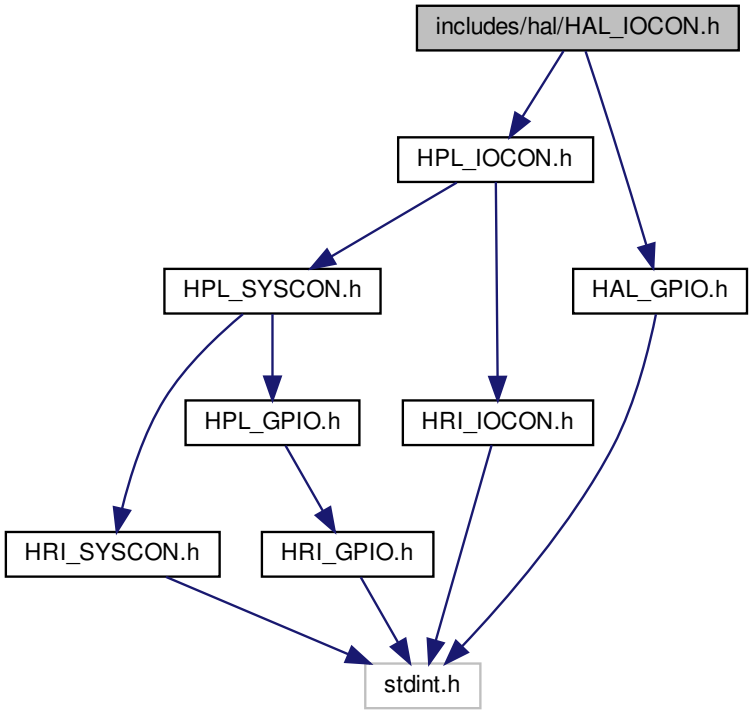
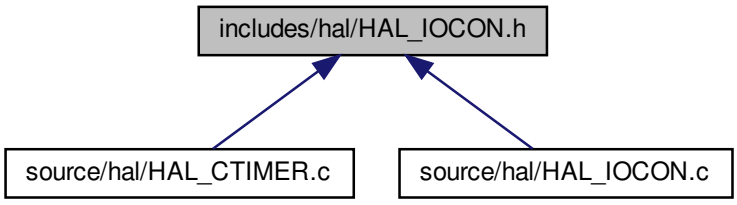


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct hal_iocon_config_t

Enumeraciones

- enum **hal_iocon_pull_mode_en** { **HAL_IOCON_PULL_NONE** = 0, **HAL_IOCON_PULL_DOWN**, **HAL_IOCON_PULL_UP**, **HAL_IOCON_PULL_REPEATER** }
- enum **hal_iocon_sample_mode_en** { **HAL_IOCON_SAMPLE_MODE_BYPASS** = 0, **HAL_IOCON_SAMPLE_MODE_1_CLOCK**, **HAL_IOCON_SAMPLE_MODE_2_CLOCK**, **HAL_IOCON_SAMPLE_MODE_3_CLOCK** }
- enum **hal_iocon_clk_sel_en** { **HAL_IOCON_CLK_DIV_0** = 0, **HAL_IOCON_CLK_DIV_1**, **HAL_IOCON_CLK_DIV_2**, **HAL_IOCON_CLK_DIV_3**, **HAL_IOCON_CLK_DIV_4**, **HAL_IOCON_CLK_DIV_5**, **HAL_IOCON_CLK_DIV_6** }
- enum **hal_iocon_iic_mode_en** { **HAL_IOCON_IIC_MODE_STANDARD** = 0, **HAL_IOCON_IIC_MODE_GPIO**, **HAL_IOCON_IIC_MODE_FAST_MODE** }

Funciones

- void hal_iocon_config_io (hal_gpio_portpin_en portpin, const hal_iocon_config_t *config)

  Configuracion de un pin.

7.5.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico IOCON (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.5.2. Documentación de las estructuras de datos

7.5.2.1. struct hal_iocon_config_t

Campos de datos

hal_iocon_pull_mode_en	pull_mode	
uint8_t	hysteresis	
uint8_t	invert_input	
uint8_t	open_drain	
hal_iocon_sample_mode_en	sample_mode	
hal_iocon_clk_sel_en	clk_sel	
uint8_t	dac_mode	
hal_iocon_iic_mode_en	iic_mode	

7.5.3. Documentación de las funciones

7.5.3.1. hal_iocon_config_io()

```
void hal_iocon_config_io (
            hal_gpio_portpin_en portpin,
            const hal_iocon_config_t * config )
```

Configuracion de un pin.

Parámetros

in	*portpin*	Puerto/pin a configurar
in	*pin_config*	Puntero a estructura de configuracion del pin

7.6. Referencia del Archivo includes/hal/HAL_PININT.h

Declaraciones a nivel de aplicacion del periferico PININT (LPC845)

```
#include <stdint.h>
#include <HAL_GPIO.h>
```

Dependencia gráfica adjunta para HAL_PININT.h:

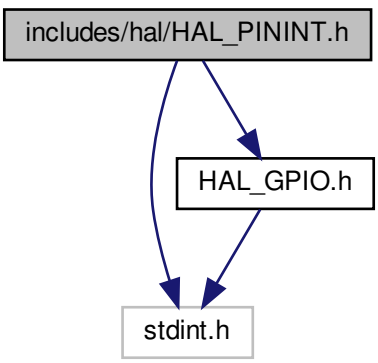
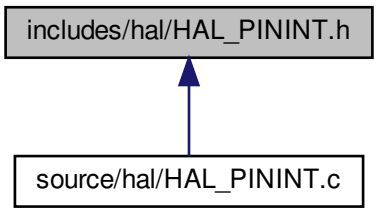


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct hal_pinint_config_t

Enumeraciones

- enum **hal_pinint_channel_en** { **HAL_PININT_CHANNEL_0** = 0, **HAL_PININT_CHANNEL_1**, **HAL_PININT_CHANNEL_2**, **HAL_PININT_CHANNEL_3**, **HAL_PININT_CHANNEL_4**, **HAL_PININT_CHANNEL_5**, **HAL_PININT_CHANNEL_6**, **HAL_PININT_CHANNEL_7** }
- enum **hal_pinint_interrupt_mode_en** { **HAL_PININT_INTERRUPT_MODE_EDGE** = 0, **HAL_PININT_INTERRUPT_MODE_LEVEL** }
- enum **hal_pinint_level_int_en** { **HAL_PININT_LEVEL_INT_HIGH** = 0, **HAL_PININT_LEVEL_INT_LOW** }

Funciones

- void hal_pinint_init (void)

  Inicializacion del modulo.
- void hal_pinint_configure_pin_interrupt (const hal_pinint_config_t *config)

  Configurar interrupciones de pin.
- void hal_pinint_register_callback (hal_pinint_channel_en channel, void(*new_callback)(void))

  Registrar callback a llamar en interrupcion de PININTn.

7.6.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico PININT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.6.2. Documentación de las funciones

7.6.2.1. hal_pinint_configure_pin_interrupt()

```
void hal_pinint_configure_pin_interrupt (
            const hal_pinint_config_t * config )
```

Configurar interrupciones de pin.

Parámetros

in	*config*	Configuracion de interrupciones de pin

7.6.2.2. hal_pinint_register_callback()

```
void hal_pinint_register_callback (
            hal_pinint_channel_en channel,
            void(*)(void) new_callback )
```

Registrar callback a llamar en interrupcion de PININTn.

Parámetros

in	*channel*	Canal al cual registrar el callback
in	*new_callback*	Puntero a funcion a ejecutar

7.7. Referencia del Archivo includes/hal/HAL_SPI.h

Declaraciones a nivel de aplicacion del periferico SPI (LPC845)

```
#include <stdint.h>
#include <HAL_SYSCON.h>
#include <HAL_GPIO.h>
```

Dependencia gráfica adjunta para HAL_SPI.h:

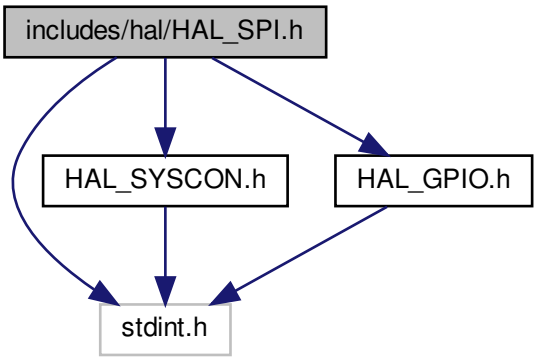
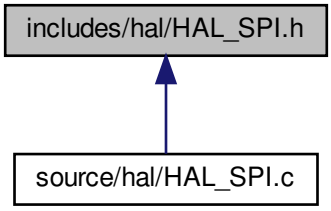


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct hal_spi_master_mode_config_t
- struct hal_spi_master_mode_tx_config_t
- struct hal_spi_master_mode_tx_data_t

defines

- #define **HAL_SPI_DUMMY_BYTE** (0xFF)

Enumeraciones

- enum **hal_spi_sel_en** { **HAL_SPI_0** = 0, **HAL_SPI_1** }
- enum **hal_spi_data_length_en** {
  HAL_SPI_DATA_LENGTH_1_BIT = 0, **HAL_SPI_DATA_LENGTH_2_BIT**, **HAL_SPI_DATA_LENGTH_3_BIT**, **HAL_SPI_DATA_LENGTH_4_BIT**,
  HAL_SPI_DATA_LENGTH_5_BIT, **HAL_SPI_DATA_LENGTH_6_BIT**, **HAL_SPI_DATA_LENGTH_7_BIT**, **HAL_SPI_DATA_LENGTH_8_BIT**,
  HAL_SPI_DATA_LENGTH_9_BIT, **HAL_SPI_DATA_LENGTH_10_BIT**, **HAL_SPI_DATA_LENGTH_11_BIT**, **HAL_SPI_DATA_LENGTH_12_BIT**,
  HAL_SPI_DATA_LENGTH_13_BIT, **HAL_SPI_DATA_LENGTH_14_BIT**, **HAL_SPI_DATA_LENGTH_15_BIT**, **HAL_SPI_DATA_LENGTH_16_BIT** }
- enum **hal_spi_clock_mode_en** { **HAL_SPI_CLOCK_MODE_0** = 0, **HAL_SPI_CLOCK_MODE_1**, **HAL_SPI_CLOCK_MODE_2**, **HAL_SPI_CLOCK_MODE_3** }
- enum **hal_spi_ssel_polarity_en** { **HAL_SPI_SSEL_POLARITY_LOW** = 0, **HAL_SPI_SSEL_POLARITY_HIGH** }
- enum **hal_spi_ssel_sel_en** {
  HAL_SPI_SSEL_SELECTION_0 = 0, **HAL_SPI_SSEL_SELECTION_1**, **HAL_SPI_SSEL_SELECTION_2**, **HAL_SPI_SSEL_SELECTION_3**,
  HAL_SPI_SSEL_SELECTION_OTHER }

Funciones

- void hal_spi_master_mode_init (hal_spi_sel_en inst, const hal_spi_master_mode_config_t *config)
  Inicializar SPI en modo master.
- uint16_t hal_spi_master_mode_rx_data (hal_spi_sel_en inst)
  Leer el dato recibido.
- void hal_spi_master_mode_config_tx (hal_spi_sel_en inst, const hal_spi_master_mode_tx_config_t *config)
  Configurar la transmision.
- void hal_spi_master_mode_tx_data (hal_spi_sel_en inst, const hal_spi_master_mode_tx_data_t *data)
  Transmitir dato.
- void hal_spi_master_mode_register_tx_callback (hal_spi_sel_en inst, void(*new_callback)(void))
  Actualizar callback en TXRDY.
- void hal_spi_master_mode_register_rx_callback (hal_spi_sel_en inst, void(*new_callback)(void))
  Actualizar callback en RXRDY.

7.7.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico SPI (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.7.2. Documentación de las estructuras de datos

7.7.2.1. struct hal_spi_master_mode_tx_config_t

Campos de datos

hal_spi_clock_mode_en	clock_mode	
uint16_t	clock_div	

7.7.2.2. struct hal_spi_master_mode_tx_data_t

Campos de datos

uint32_t	data: 16	
uint32_t	ssel0_n: 1	
uint32_t	ssel1_n: 1	
uint32_t	ssel2_n: 1	
uint32_t	ssel3_n: 1	
uint32_t	eot: 1	
uint32_t	eof: 1	
uint32_t	rxignore: 1	
uint32_t	__pad0__: 1	
uint32_t	data_length: 4	
uint32_t	__pad1__: 4	

7.7.3. Documentación de las funciones

7.7.3.1. hal_spi_master_mode_init()

```
void hal_spi_master_mode_init (
            hal_spi_sel_en inst,
            const hal_spi_master_mode_config_t * config )
```

Inicializar SPI en modo master.

Parámetros

in	*inst*	Instancia de SPI a inicializar
in	*config*	Configuracion deseada

7.7.3.2. hal_spi_master_mode_rx_data()

```
uint16_t hal_spi_master_mode_rx_data (
            hal_spi_sel_en inst )
```

Leer el dato recibido.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Dato recibido

7.7.3.3. hal_spi_master_mode_config_tx()

```
void hal_spi_master_mode_config_tx (
            hal_spi_sel_en inst,
            const hal_spi_master_mode_tx_config_t * config )
```

Configurar la transmision.

Parámetros

in	*inst*	Instancia a configurar
in	*config*	Configuracion para la transmision deseada

7.7.3.4. hal_spi_master_mode_tx_data()

```
void hal_spi_master_mode_tx_data (
            hal_spi_sel_en inst,
            const hal_spi_master_mode_tx_data_t * data )
```

Transmitir dato.

Parámetros

in	*inst*	Instancia a utilizar
in	*data*	Dato a transmitir, con controles asociados

7.7.3.5. hal_spi_master_mode_register_tx_callback()

```
void hal_spi_master_mode_register_tx_callback (
            hal_spi_sel_en inst,
            void(*)(void) new_callback )
```

Actualizar callback en TXRDY.

Parámetros

in	*inst*	Instancia a configurar
in	*new_callback*	Nuevo callback a ejecutar en TXRDY

7.7.3.6. hal_spi_master_mode_register_rx_callback()

```
void hal_spi_master_mode_register_rx_callback (
            hal_spi_sel_en inst,
            void(*)(void) new_callback )
```

Actualizar callback en RXRDY.

Parámetros

in	*inst*	Instancia a configurar
in	*new_callback*	Nuevo callback a ejecutar en RXRDY

7.8. Referencia del Archivo includes/hal/HAL_SYSCON.h

Declaraciones a nivel de aplicacion del periferico SYSCON (LPC845)

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

Dependencia gráfica adjunta para HAL_SYSCON.h:

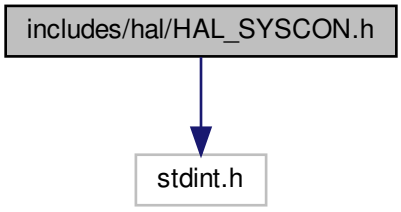
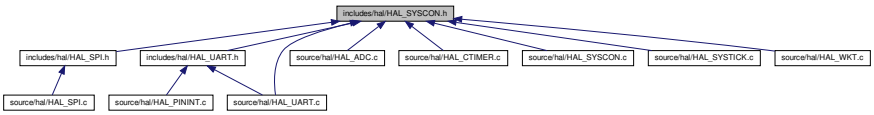


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **hal_syscon_clkout_source_sel_en** { **HAL_SYSCON_CLKOUT_SOURCE_SEL_FRO** = 0, **HAL_SYSCON_CLKOUT_SOURCE_SEL_MAIN_CLOCK**, **HAL_SYSCON_CLKOUT_SOURCE_SEL_SYS_PLL**, **HAL_SYSCON_CLKOUT_SOURCE_SEL_EXT_CLOCK**, **HAL_SYSCON_CLKOUT_SOURCE_SEL_WATCHDOG_OSC** }
- enum **hal_syscon_frg_clock_sel_en** { **HAL_SYSCON_FRG_CLOCK_SEL_FRO** = 0, **HAL_SYSCON_FRG_CLOCK_SEL_MAIN_CLOCK**, **HAL_SYSCON_FRG_CLOCK_SEL_SYS_PLL**, **HAL_SYSCON_FRG_CLOCK_SEL_NONE** }
- enum **hal_syscon_peripheral_sel_en** { **HAL_SYSCON_PERIPHERAL_SEL_UART0** = 0, **HAL_SYSCON_PERIPHERAL_SEL_UART1**, **HAL_SYSCON_PERIPHERAL_SEL_UART2**, **HAL_SYSCON_PERIPHERAL_SEL_UART3**, **HAL_SYSCON_PERIPHERAL_SEL_UART4**, **HAL_SYSCON_PERIPHERAL_SEL_IIC0**, **HAL_SYSCON_PERIPHERAL_SEL_IIC1**, **HAL_SYSCON_PERIPHERAL_SEL_IIC2**, **HAL_SYSCON_PERIPHERAL_SEL_IIC3**, **HAL_SYSCON_PERIPHERAL_SEL_SPI0**, **HAL_SYSCON_PERIPHERAL_SEL_SPI1** }
- enum **hal_syscon_peripheral_clock_sel_en** { **HAL_SYSCON_PERIPHERAL_CLOCK_SEL_FRO** = 0, **HAL_SYSCON_PERIPHERAL_CLOCK_SEL_MAIN**, **HAL_SYSCON_PERIPHERAL_CLOCK_SEL_FRG0**, **HAL_SYSCON_PERIPHERAL_CLOCK_SEL_FRG1**, **HAL_SYSCON_PERIPHERAL_CLOCK_SEL_FRO_DIV**, **HAL_SYSCON_PERIPHERAL_CLOCK_SEL_NONE** = 7 }
- enum **hal_syscon_iocon_glitch_sel_en** { **HAL_SYSCON_IOCON_GLITCH_SEL_0** = 0, **HAL_SYSCON_IOCON_GLITCH_SEL_1**, **HAL_SYSCON_IOCON_GLITCH_SEL_2**, **HAL_SYSCON_IOCON_GLITCH_SEL_3**, **HAL_SYSCON_IOCON_GLITCH_SEL_4**, **HAL_SYSCON_IOCON_GLITCH_SEL_5**, **HAL_SYSCON_IOCON_GLITCH_SEL_6**, **HAL_SYSCON_IOCON_GLITCH_SEL_7** }
- enum **hal_syscon_pll_source_sel_en** { **HAL_SYSCON_PLL_SOURCE_SEL_FRO** = 0, **HAL_SYSCON_PLL_SOURCE_SEL_EXT_CLK**, **HAL_SYSCON_PLL_SOURCE_SEL_WATCHDOG**, **HAL_SYSCON_PLL_SOURCE_SEL_FRO_DIV** }

Funciones

- uint32_t hal_syscon_get_system_clock (void)

  Obtener la frecuencia actual del main clock.
- uint32_t hal_syscon_get_fro_clock (void)

  Obtener la frecuencia actual del FRO.
- void hal_syscon_config_external_crystal (uint32_t crystal_freq, uint8_t use_as_main)

  Configurar el ext clock a partir de un cristal externo.
- void hal_syscon_config_fro_direct (uint8_t direct, uint8_t use_as_main)

  Configurar el clock FRO.
- void hal_syscon_config_clkout (uint8_t port, uint8_t pin, hal_syscon_clkout_source_sel_en clock_source, uint8_t divider)

  Configurar el pin de clock out (salida de clock hacia afuera)
- void hal_syscon_config_frg (uint8_t inst, hal_syscon_frg_clock_sel_en clock_source, uint32_t mul)

  Configurar el divisor fraccional.
- void hal_syscon_set_peripheral_clock_source (hal_syscon_peripheral_sel_en peripheral, hal_syscon_peripheral_clock_sel_en clock_source)

  Fijar la fuente de clock de un periferico.
- uint32_t hal_syscon_get_peripheral_clock (hal_syscon_peripheral_sel_en peripheral)

  Obtener la frecuencia de clock en Hz configurada para cierto periferico.
- void hal_syscon_set_iocon_glitch_divider (hal_syscon_iocon_glitch_sel_en sel, uint32_t div)

  Configurar divisor para el clock de glitches del IOCON.

- void hal_syscon_config_pll (hal_syscon_pll_source_sel_en clock_source, uint32_t freq)
  Configurar el PLL.
- uint32_t hal_syscon_get_pll_clock (void)
  Obtener frecuencia actual configurada del PLL.

7.8.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico SYSCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.8.2. Documentación de las funciones

7.8.2.1. hal_syscon_get_system_clock()

```
uint32_t hal_syscon_get_system_clock (
            void  )
```

Obtener la frecuencia actual del main clock.

Devuelve

Frecuencia del main clock en Hz

7.8.2.2. hal_syscon_get_fro_clock()

```
uint32_t hal_syscon_get_fro_clock (
            void  )
```

Obtener la frecuencia actual del FRO.

Devuelve

Frecuencia del FRO en Hz

7.8.2.3. hal_syscon_config_external_crystal()

```
void hal_syscon_config_external_crystal (
            uint32_t crystal_freq,
            uint8_t use_as_main )
```

Configurar el ext clock a partir de un cristal externo.

Parámetros

in	*crystal_freq*	Frecuencia del cristal externo utilizado
in	*use_as_main*	Si es distinto de cero, se utilizara el oscilador a cristal como main clock

7.8.2.4. hal_syscon_config_fro_direct()

```
void hal_syscon_config_fro_direct (
            uint8_t direct,
            uint8_t use_as_main )
```

Configurar el clock FRO.

Parámetros

in	*direct*	Si es distinto de cero se omite el divisor del FRO
in	*use_as_main*	Si es distinto de cero, se utilizara el FRO como main clock

7.8.2.5. hal_syscon_config_clkout()

```
void hal_syscon_config_clkout (
            uint8_t port,
            uint8_t pin,
            hal_syscon_clkout_source_sel_en clock_source,
            uint8_t divider )
```

Configurar el pin de clock out (salida de clock hacia afuera)

Parámetros

in	*port*	Numero de puerto por donde sacar el clock out
in	*pin*	Numero de pin por donde sacar el clock out
in	*clock_source*	Fuente deseada para la salida clock out
in	*divider*	Divisor deseado para la salida clock out

7.8.2.6. hal_syscon_config_frg()

```
void hal_syscon_config_frg (
            uint8_t inst,
            hal_syscon_frg_clock_sel_en clock_source,
            uint32_t mul )
```

Configurar el divisor fraccional.

El divisor siempre se debe fijar en 256 para estos MCU.

Parámetros

in	*inst*	Instancia de FRG a configurar
in	*clock_source*	Fuente de clock de entrada para el FRG
in	*mul*	Multiplicador deseado

7.8.2.7. hal_syscon_set_peripheral_clock_source()

```
void hal_syscon_set_peripheral_clock_source (
            hal_syscon_peripheral_sel_en peripheral,
            hal_syscon_peripheral_clock_sel_en clock_source )
```

Fijar la fuente de clock de un periferico.

Parámetros

in	*peripheral*	Periferico deseado
in	*clock_source*	Fuente de clock deseada

7.8.2.8. hal_syscon_get_peripheral_clock()

```
uint32_t hal_syscon_get_peripheral_clock (
            hal_syscon_peripheral_sel_en peripheral )
```

Obtener la frecuencia de clock en Hz configurada para cierto periferico.

Parámetros

in	*peripheral*	Periferico deseado

Devuelve

Frecuencia en Hz del clock del periferico

7.8.2.9. hal_syscon_set_iocon_glitch_divider()

```
void hal_syscon_set_iocon_glitch_divider (
            hal_syscon_iocon_glitch_sel_en sel,
            uint32_t div )
```

Configurar divisor para el clock de glitches del IOCON.

Parámetros

in	*sel*	Seleccion de divisor
in	*div*	Valor de division deseado

7.8.2.10. hal_syscon_config_pll()

```
void hal_syscon_config_pll (
            hal_syscon_pll_source_sel_en clock_source,
            uint32_t freq )
```

Configurar el PLL.

Parámetros

in	*clock_source*	Fuente de clock de referencia para el PLL
in	*freq*	Frecuencia deseada de salida del PLL

7.8.2.11. hal_syscon_get_pll_clock()

```
uint32_t hal_syscon_get_pll_clock (
            void  )
```

Obtener frecuencia actual configurada del PLL.

Devuelve

Frecuencia actual del PLL en Hz

7.9. Referencia del Archivo includes/hal/HAL_SYSTICK.h

Declaraciones a nivel de aplicacion del periferico SYSICK (LPC845)

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

Dependencia gráfica adjunta para HAL_SYSTICK.h:

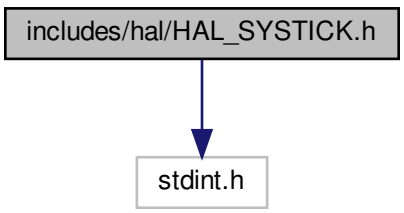
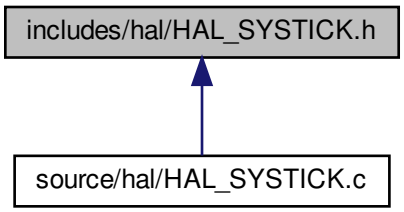


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Funciones

- void hal_systick_init (uint32_t tick_us, void(∗callback)(void))
  Inicializacion del SYSTICK.
- void hal_systick_update_callback (void(∗callback)(void))
  Actualizar callback del SYSTICK.

7.9.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico SYSICK (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.9.2. Documentación de las funciones

7.9.2.1. hal_systick_init()

```
void hal_systick_init (
            uint32_t tick_us,
            void(*)(void) callback )
```

Inicializacion del SYSTICK.

Parámetros

in	*tick_us*	Tiempo en microsegundos deseado para el tick
in	*callback*	Funcion a llamar en cada tick

Ejemplos:

Ejemplo_ADC.c.

7.9.2.2. hal_systick_update_callback()

```
void hal_systick_update_callback (
            void(*)(void) callback )
```

Actualizar callback del SYSTICK.

Parámetros

in	*callback*	Nuevo callback a ejecutar en cada tick

7.10. Referencia del Archivo includes/hal/HAL_UART.h

Declaraciones a nivel de aplicacion del periferico UART (LPC845)

```
#include <stdint.h>
#include <HAL_SYSCON.h>
#include <HAL_GPIO.h>
```

Dependencia gráfica adjunta para HAL_UART.h:

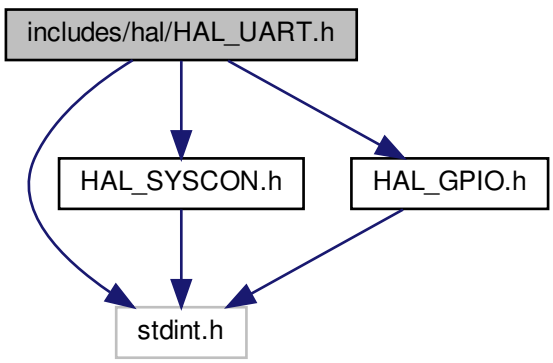
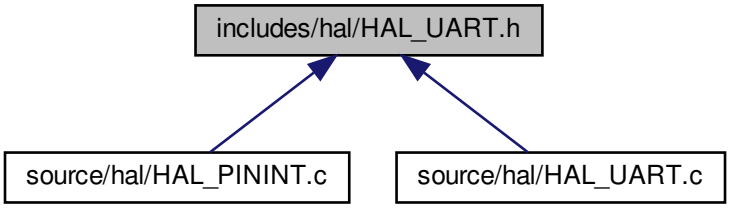


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct hal_uart_config_t

Enumeraciones

- enum **hal_uart_datalen_en** { **HAL_UART_DATALEN_7BIT** = 0, **HAL_UART_DATALEN_8BIT**, **HAL_U↩ ART_DATALEN_9BIT** }
- enum **hal_uart_parity_en** { **HAL_UART_PARITY_NO_PARITY** = 0, **HAL_UART_PARITY_EVEN** = 2, **H↩ AL_UART_PARITY_ODD** }
- enum **hal_uart_stop_en** { **HAL_UART_STOPLEN_1BIT** = 0, **HAL_UART_STOPLEN_2BIT** }
- enum **hal_uart_oversampling_en** { **HAL_UART_OVERSAMPLING_X5** = 4, **HAL_UART_OVERSAMPLING_X6**, **HAL_UART_OVERSAMPL↩ ING_X7**, **HAL_UART_OVERSAMPLING_X8**, **HAL_UART_OVERSAMPLING_X9**, **HAL_UART_OVERSAMPLING_X10**, **HAL_UART_OVERSAMPLIN↩ G_X11**, **HAL_UART_OVERSAMPLING_X12**, **HAL_UART_OVERSAMPLING_X13**, **HAL_UART_OVERSAMPLING_X14**, **HAL_UART_OVERSAMPLI↩ NG_X15**, **HAL_UART_OVERSAMPLING_X16** }
- enum **hal_uart_tx_result** { **HAL_UART_TX_RESULT_OK** = 0, **HAL_UART_TX_RESULT_NOT_READY** }
- enum **hal_uart_rx_result** { **HAL_UART_RX_RESULT_OK** = 0, **HAL_UART_RX_RESULT_NOT_READY** }

Funciones

- void hal_uart_init (uint8_t inst, const hal_uart_config_t ∗config)

  Inicializar UART con los parametros deseados.
- hal_uart_tx_result hal_uart_tx_byte (uint8_t inst, uint32_t data)

  Transmitir un dato mediante la UART.
- hal_uart_rx_result hal_uart_rx_byte (uint8_t inst, uint32_t ∗data)

  Recibir un dato de la UART.
- void hal_uart_register_tx_callback (uint8_t inst, void(∗new_callback)(void))

  Registrar el callback a ser llamado una vez finalizada la transmision de un dato por UART.
- void hal_uart_register_rx_callback (uint8_t inst, void(∗new_callback)(void))

  Registrar el callback a ser llamado en la recepcion de un dato por UART.
- void UART3_irq (void)

  Interrupcion de UART3.
- void UART4_irq (void)

  Interrupcion de UART4.

7.10.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico UART (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.10.2. Documentación de las funciones

7.10.2.1. hal_uart_init()

```
void hal_uart_init (
            uint8_t inst,
            const hal_uart_config_t * config )
```

Inicializar UART con los parametros deseados.

Parámetros

in	*inst*	Que instancia de UART inicializar
in	*config*	Puntero a configuracion de la UART

7.10.2.2. hal_uart_tx_byte()

```
hal_uart_tx_result hal_uart_tx_byte (
            uint8_t inst,
            uint32_t data )
```

Transmitir un dato mediante la UART.

Parámetros

in	*inst*	Que instancia de UART usar
in	*data*	Dato a transmitir. Puede ser de 7, 8 o 9 bits

7.10.2.3. hal_uart_rx_byte()

```
hal_uart_rx_result hal_uart_rx_byte (
            uint8_t inst,
            uint32_t * data )
```

Recibir un dato de la UART.

Parámetros

in	*inst*	Que instancia de UART usar
in	*data*	Puntero a donde guardar el dato recibido

Devuelve

Estado de la recepcion

7.10.2.4. hal_uart_register_tx_callback()

```
void hal_uart_register_tx_callback (
            uint8_t inst,
            void(*)(void) new_callback )
```

Registrar el callback a ser llamado una vez finalizada la transmision de un dato por UART.

Parámetros

in	*inst*	A que instancia de UART registrar el callback
in	*new_callback*	Puntero a funcion a llamar cada vez que se termina de enviar un dato por UART

7.10.2.5. hal_uart_register_rx_callback()

```
void hal_uart_register_rx_callback (
            uint8_t inst,
            void(*)(void) new_callback )
```

Registrar el callback a ser llamado en la recepcion de un dato por UART.

Parámetros

in	*inst*	A que instancia de UART registrar el callback
in	*new_callback*	Puntero a funcion a llamar cada vez que se recibe un dato por UART

7.11. Referencia del Archivo includes/hal/HAL_WKT.h

Declaraciones a nivel de aplicacion del periferico WKT (LPC845)

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

Dependencia gráfica adjunta para HAL_WKT.h:

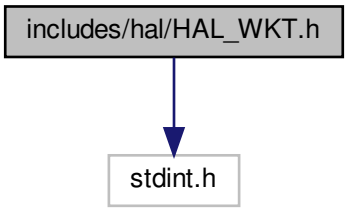
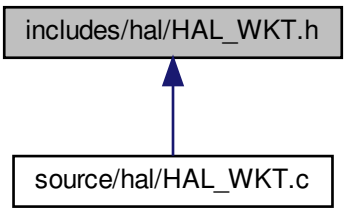


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **hal_wkt_clock_source_en** { **HAL_WKT_CLOCK_SOURCE_FRO_DIV** = 0, **HAL_WKT_CLOCK_SOURCE_LOW_POWER_OSC**, **HAL_WKT_CLOCK_SOURCE_EXTERNAL** }

Funciones

- void hal_wkt_init (hal_wkt_clock_source_en clock_sel, uint32_t ext_clock_value, void(∗callback)(void))
  Inicializar el WKT.
- void **hal_wkt_select_clock_source** (hal_wkt_clock_source_en clock_sel, uint32_t ext_clock_value)
- void hal_wkt_register_callback (void(∗new_callback)(void))
  Registrar un callback para la interrupcion del WKT.
- void **hal_wkt_start_count** (uint32_t time_useg)
- void **hal_wkt_start_count_with_value** (uint32_t value)

7.11.1. Descripción detallada

Declaraciones a nivel de aplicacion del periferico WKT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.11.2. Documentación de las funciones

7.11.2.1. hal_wkt_init()

```
void hal_wkt_init (
            hal_wkt_clock_source_en clock_sel,
            uint32_t ext_clock_value,
            void(*)(void) callback )
```

Inicializar el WKT.

Parámetros

in	*clock_sel*	Seleccion de clock deseada para el WKT
in	*ext_clock_value*	Valor de clock externo (si la seleccion es interna, no importa este parametro)
in	*callback*	Callback a ejecutar en la interrupcion del WKT

7.11.2.2. hal_wkt_register_callback()

```
void hal_wkt_register_callback (
            void(*)(void) new_callback )
```

Registrar un callback para la interrupcion del WKT.

Parámetros

in	*new_callback*	Nuevo callback para la interrupcion del WKT

7.12. Referencia del Archivo includes/hpl/HPL_ADC.h

Declaraciones a nivel de abstraccion de periferico del ADC (LPC845)

```
#include <HRI_ADC.h>
```

Dependencia gráfica adjunta para HPL_ADC.h:

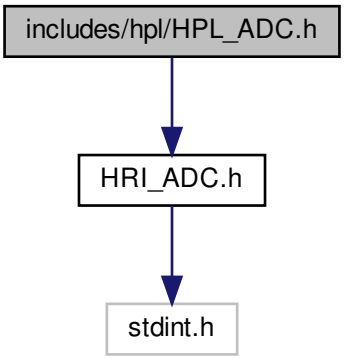
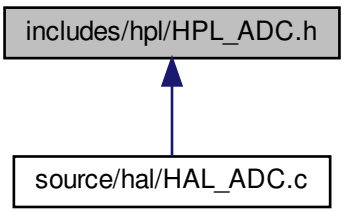


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct ADC_global_data_t
- struct ADC_channel_data_t

Enumeraciones

- enum **ADC_sequence_sel_en** { **ADC_SEQUENCE_SEL_A** = 0, **ADC_SEQUENCE_SEL_B** }
- enum **ADC_operation_mode_en** { **ADC_OPERATION_MODE_SYNCHRONOUS** = 0, **ADC_OPERATIO↩ N_MODE_ASYNCHRONOUS** }

- enum **ADC_low_power_mode_en** { **ADC_LOW_POWER_MODE_DISABLED** = 0, **ADC_LOW_POWER_MODE_ENABLED** }
- enum **ADC_trigger_sel_en** { **ADC_TRIGGER_SEL_NONE** = 0, **ADC_TRIGGER_SEL_PININT0_IRQ**, **ADC_TRIGGER_SEL_PININT1_IRQ**, **ADC_TRIGGER_SEL_SCT0_OUT3**, **ADC_TRIGGER_SEL_SCT0_OUT4**, **ADC_TRIGGER_SEL_T0_MAT3**, **ADC_TRIGGER_SEL_CMP0_OUT_ADC**, **ADC_TRIGGER_SEL_GPIO_INT_BMAT**, **ADC_TRIGGER_SEL_ARM_TXEV** }
- enum **ADC_trigger_pol_sel_en** { **ADC_TRIGGER_POL_SEL_NEGATIVE_EDGE** = 0, **ADC_TRIGGER_POL_SEL_POSITIVE_EDGE** }
- enum **ADC_sync_sel_en** { **ADC_SYNC_SEL_ENABLE_SYNC** = 0, **ADC_SYNC_SEL_BYPASS_SYNC** }
- enum **ADC_clock_source_en** { **ADC_CLOCK_SOURCE_FRO** = 0, **ADC_CLOCK_SOURCE_PLL** }
- enum **ADC_threshold_sel_en** { **ADC_THRESHOLD_SEL_0** = 0, **ADC_THRESHOLD_SEL_1** }
- enum **ADC_interrupt_mode_en** { **ADC_INTERRUPT_MODE_EOC** = 0, **ADC_INTERRUPT_MODE_EOS** }
- enum **ADC_threshold_interrupt_sel_en** { **ADC_THRESHOLD_INTERRUPT_SEL_OUTSIDE** = 1, **ADC_THRESHOLD_INTERRUPT_SEL_CROSSING** }
- enum **ADC_vrange_sel_en** { **ADC_VRANGE_HIGH_VOLTAGE** = 0, **ADC_VRANGE_LOW_VOLTAGE** }

Funciones

- static void ADC_control_config (uint8_t div, ADC_operation_mode_en operation, ADC_low_power_mode_en power)

  Configuracion del registro de control del ADC.
- static void ADC_sequence_config_channels (ADC_sequence_sel_en sequence, uint16_t channels)

  Configuracion de canales habilitados en una secuencia.
- static uint16_t ADC_sequence_get_channels (ADC_sequence_sel_en sequence)

  Obtener los canales configuados en la secuencia.
- static void ADC_sequence_config_trigger (ADC_sequence_sel_en sequence, ADC_trigger_sel_en trigger)

  Configuracion de trigger en una secuencia.
- static void ADC_sequence_config_trigger_pol (ADC_sequence_sel_en sequence, ADC_trigger_pol_sel_en pol)

  Configuracion de la polaridad del trigger en una secuencia.
- static void ADC_sequence_config_sync (ADC_sequence_sel_en sequence, ADC_sync_sel_en sync)

  Configuracion de la sincronizacion en una secuencia.
- static void ADC_sequence_set_start (ADC_sequence_sel_en sequence)

  Iniciar conversiones por software en una secuencia.
- static void ADC_sequence_set_burst (ADC_sequence_sel_en sequence)

  Iniciar conversiones en rafaga en una secuencia.
- static void ADC_sequence_clear_burst (ADC_sequence_sel_en sequence)

  Detener conversiones en rafaga en una secuencia.
- static void ADC_sequence_set_singlestep (ADC_sequence_sel_en sequence)

  Fijar modo singlestep.
- static void ADC_sequence_clear_singlestep (ADC_sequence_sel_en sequence)

  Limpiar modo singlestep.
- static void ADC_sequence_A_make_low_priority (void)

  Fijar baja prioridad para la secuencia A.
- static void ADC_sequence_A_make_high_priority (void)

  Fijar alta prioridad para la secuencia A.
- static void ADC_sequence_config_interrupt_mode (ADC_sequence_sel_en sequence, ADC_interrupt_mode_en mode)

  Configurar modo de interrupcion para una secuencia.
- static ADC_interrupt_mode_en ADC_sequence_get_mode (ADC_sequence_sel_en sequence)

Obtener modo de interrupcion de una secuencia.

- static void ADC_sequence_enable (ADC_sequence_sel_en sequence)

  Habilitar secuencia.
- static void ADC_sequence_disable (ADC_sequence_sel_en sequence)

  Inhabilitar secuencia.
- static void ADC_set_compare_low_threshold (ADC_threshold_sel_en threshold_selection, uint16_t threshold_value)

  Fijar valor de umbral de comparacion (parte baja)
- static void ADC_set_compare_high_threshold (ADC_threshold_sel_en threshold_selection, uint16_t threshold_value)

  Fijar valor de umbral de comparacion (parte alta)
- static void ADC_set_channel_threshold (uint8_t channel, ADC_threshold_sel_en threshold_selection)

  Fijar contra que umbral de comparacion se compara el canal.
- static void ADC_enable_sequence_interrupt (ADC_sequence_sel_en sequence)

  Habilitacion de interrupcion de secuencia.
- static void ADC_disable_sequence_interrupt (ADC_sequence_sel_en sequence)

  Inhabilitacion de interrupcion de secuencia.
- static void ADC_enable_overrun_interrupt (void)

  Habilitacion de interrupcion de overrun.
- static void ADC_disable_overrun_interrupt (void)

  Inhabilitacion de interrupcion de overrun.
- static void ADC_enable_threshold_interrupt (uint8_t channel, ADC_threshold_interrupt_sel_en mode)

  Habilitacion de interrupcion de threshold.
- static void ADC_disable_threshold_interrupt (uint8_t channel)

  Inhabilitacion de interrupcion de threshold.
- static ADC_global_data_t ADC_get_global_data (ADC_sequence_sel_en sequence)

  Leer alguno de los registros globales de resultado de conversion.
- static ADC_channel_data_t ADC_get_channel_data (uint8_t channel)

  Leer alguno de los registros de canal de resultado de conversion.
- static void ADC_set_vrange (ADC_vrange_sel_en vrange)

  Configuracion de rango de tension.
- static void ADC_hardware_calib (uint8_t div)

  Realizacion de calibracion de hardware.

Variables

- volatile ADC_per_t *const ADC

  Periferico ADC.

7.12.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del ADC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.12.2. Documentación de las estructuras de datos

7.12.2.1. struct ADC_global_data_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	RESULT: 12	
uint32_t	THCMPRANGE: 2	
uint32_t	THCMPCROSS: 2	
uint32_t	__pad1__: 6	
uint32_t	CHANNEL: 4	
uint32_t	OVERRUN: 1	
uint32_t	DATAVALID: 1	

7.12.2.2. struct ADC_channel_data_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	RESULT: 12	
uint32_t	THCMPRANGE: 2	
uint32_t	THCMPCROSS: 2	
uint32_t	__pad1__: 6	
uint32_t	CHANNEL: 4	
uint32_t	OVERRUN: 1	
uint32_t	DATAVALID: 1	

7.12.3. Documentación de las funciones

7.12.3.1. ADC_control_config()

```
static void ADC_control_config (
            uint8_t div,
            ADC_operation_mode_en operation,
            ADC_low_power_mode_en power )  [inline], [static]
```

Configuracion del registro de control del ADC.

Parámetros

in	*div*	Divisor deseado
in	*operation*	Modo de operacion sincronico/asincronico
in	*power*	Modo de consumo

7.12.3.2. ADC_sequence_config_channels()

```
static void ADC_sequence_config_channels (
            ADC_sequence_sel_en sequence,
            uint16_t channels )  [inline], [static]
```

Configuracion de canales habilitados en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar
in	*channels*	Canales a habilitar en forma de mascara

7.12.3.3. ADC_sequence_get_channels()

```
static uint16_t ADC_sequence_get_channels (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Obtener los canales configuados en la secuencia.

Parámetros

in	*sequence*	Secuencia a consultar

Devuelve

Mascara de canales habilitados en la secuencia

7.12.3.4. ADC_sequence_config_trigger()

```
static void ADC_sequence_config_trigger (
            ADC_sequence_sel_en sequence,
            ADC_trigger_sel_en trigger )  [inline], [static]
```

Configuracion de trigger en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar
in	*trigger*	Fuente de trigger deseada

7.12.3.5. ADC_sequence_config_trigger_pol()

```
static void ADC_sequence_config_trigger_pol (
            ADC_sequence_sel_en sequence,
            ADC_trigger_pol_sel_en pol )  [inline], [static]
```

Configuracion de la polaridad del trigger en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar
in	*pol*	Polaridad del trigger deseado

7.12.3.6. ADC_sequence_config_sync()

```
static void ADC_sequence_config_sync (
            ADC_sequence_sel_en sequence,
            ADC_sync_sel_en sync )  [inline], [static]
```

Configuracion de la sincronizacion en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar
in	*sync*	Metodo de sincronizacion deseado

7.12.3.7. ADC_sequence_set_start()

```
static void ADC_sequence_set_start (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Iniciar conversiones por software en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar

7.12.3.8. ADC_sequence_set_burst()

```
static void ADC_sequence_set_burst (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Iniciar conversiones en rafaga en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar

7.12.3.9. ADC_sequence_clear_burst()

```
static void ADC_sequence_clear_burst (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Detener conversiones en rafaga en una secuencia.

Parámetros

in	*sequence*	Secuencia a configurar

7.12.3.10. ADC_sequence_set_singlestep()

```
static void ADC_sequence_set_singlestep (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Fijar modo singlestep.

Parámetros

in	*sequence*	Secuencia a configurar

7.12.3.11. ADC_sequence_clear_singlestep()

```
static void ADC_sequence_clear_singlestep (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Limpiar modo singlestep.

Parámetros

in	*sequence*	Secuencia a configurar

7.12.3.12. ADC_sequence_config_interrupt_mode()

```
static void ADC_sequence_config_interrupt_mode (
            ADC_sequence_sel_en sequence,
            ADC_interrupt_mode_en mode )  [inline], [static]
```

Configurar modo de interrupcion para una secuencia.

Parámetros

in	*sequence*	Que secuencia configurar
in	*mode*	Modo de interrupcion

7.12.3.13. ADC_sequence_get_mode()

```
static ADC_interrupt_mode_en ADC_sequence_get_mode (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Obtener modo de interrupcion de una secuencia.

Parámetros

in	*sequence*	Secuencia a consultar

Devuelve

Modo de interrupcion de la secuencia

7.12.3.14. ADC_sequence_enable()

```
static void ADC_sequence_enable (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Habilitar secuencia.

Parámetros

in	*sequence*	Que secuencia habilitar

7.12.3.15. ADC_sequence_disable()

```
static void ADC_sequence_disable (
```

```
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Inhabilitar secuencia.

Parámetros

in	*sequence*	Que secuencia inhabilitar

7.12.3.16. ADC_set_compare_low_threshold()

```
static void ADC_set_compare_low_threshold (
            ADC_threshold_sel_en threshold_selection,
            uint16_t threshold_value )  [inline], [static]
```

Fijar valor de umbral de comparacion (parte baja)

Parámetros

in	*threshold_selection*	Que threshold fijar
in	*threshold_value*	Valor de threshold a fijar

7.12.3.17. ADC_set_compare_high_threshold()

```
static void ADC_set_compare_high_threshold (
            ADC_threshold_sel_en threshold_selection,
            uint16_t threshold_value )  [inline], [static]
```

Fijar valor de umbral de comparacion (parte alta)

Parámetros

in	*threshold_selection*	Que threshold fijar
in	*threshold_value*	Valor de threshold a fijar

7.12.3.18. ADC_set_channel_threshold()

```
static void ADC_set_channel_threshold (
            uint8_t channel,
            ADC_threshold_sel_en threshold_selection )  [inline], [static]
```

Fijar contra que umbral de comparacion se compara el canal.

Parámetros

in	*channel*	Numero de canal
in	*threshold_selection*	Contra que umbral de comparacion comparar

7.12.3.19. ADC_enable_sequence_interrupt()

```
static void ADC_enable_sequence_interrupt (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Habilitacion de interrupcion de secuencia.

Parámetros

in	*sequence*	Sobre que secuencia habilitar la interrupcion

7.12.3.20. ADC_disable_sequence_interrupt()

```
static void ADC_disable_sequence_interrupt (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Inhabilitacion de interrupcion de secuencia.

Parámetros

in	*sequence*	Sobre que secuencia inhabilitar la interrupcion

7.12.3.21. ADC_enable_threshold_interrupt()

```
static void ADC_enable_threshold_interrupt (
            uint8_t channel,
            ADC_threshold_interrupt_sel_en mode )  [inline], [static]
```

Habilitacion de interrupcion de threshold.

Parámetros

in	*channel*	Sobre que canal habilitar la interrupcion
in	*mode*	Modo de interrupcion

7.12.3.22. ADC_disable_threshold_interrupt()

```
static void ADC_disable_threshold_interrupt (
            uint8_t channel )  [inline], [static]
```

Inhabilitacion de interrupcion de threshold.

Parámetros

in	*channel*	Sobre que canal inhabilitar la interrupcion

7.12.3.23. ADC_get_global_data()

```
static ADC_global_data_t ADC_get_global_data (
            ADC_sequence_sel_en sequence )  [inline], [static]
```

Leer alguno de los registros globales de resultado de conversion.

Parámetros

in	*sequence*	De que secuencia leer el resultado global de conversion

Devuelve

Contenido del registro

7.12.3.24. ADC_get_channel_data()

```
static ADC_channel_data_t ADC_get_channel_data (
            uint8_t channel )  [inline], [static]
```

Leer alguno de los registros de canal de resultado de conversion.

Parámetros

in	*channel*	De que canal leer el resultado de canal de conversion

Devuelve

Contenido del registro

7.12.3.25. ADC_set_vrange()

```
static void ADC_set_vrange (
            ADC_vrange_sel_en vrange )  [inline], [static]
```

Configuracion de rango de tension.

Parámetros

in	*vrange*	Rango de tension de trabajo

7.12.3.26. ADC_hardware_calib()

```
static void ADC_hardware_calib (
            uint8_t div )  [inline], [static]
```

Realizacion de calibracion de hardware.

Parámetros

in	*div*	Divisor utilizado para lograr los 500KHz de clock

7.13. Referencia del Archivo includes/hpl/HPL_CTIMER.h

Definiciones a nivel de abstraccion del periferico CTIMER (LPC845)

```
#include <HRI_CTIMER.h>
```

Dependencia gráfica adjunta para HPL_CTIMER.h:

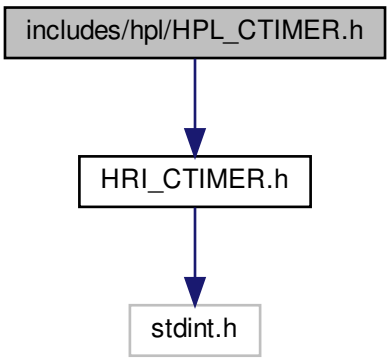
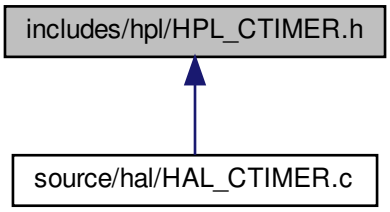


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct CTIMER_MR_config_t
- struct CTIMER_CC_config_t
- struct CTIMER_CTCR_config_t
- struct CTIMER_EMR_config_t

Enumeraciones

- enum **CTIMER_EMC_config_en** { **CTIMER_EMC_CONFIG_DO_NOTHING** = 0, **CTIMER_EMC_CONFIG_CLEAR**, **CTIMER_EMC_CONFIG_SET**, **CTIMER_EMC_CONFIG_TOGGLE** }
- enum **CTIMER_CTMODE_config_en** { **CTIMER_CTMODE_CONFIG_TIMER_MODE** = 0, **CTIMER_CTMODE_CONFIG_RISING_EDGE**, **CTIMER_CTMODE_CONFIG_FALLING_EDGE**, **CTIMER_CTMODE_CONFIG_DOUBLE_EDGE** }
- enum **CTIMER_CINSEL_config_en** { **CTIMER_CINSEL_CONFIG_CAP0** = 0, **CTIMER_CINSEL_CONFIG_CAP1**, **CTIMER_CINSEL_CONFIG_CAP2**, **CTIMER_CINSEL_CONFIG_CAP3** }
- enum **CTIMER_SELCC_config_en** { **CTIMER_SELCC_CONFIG_CH0_RISING** = 0, **CTIMER_SELCC_CONFIG_CH0_FALLING**, **CTIMER_SELCC_CONFIG_CH1_RISING**, **CTIMER_SELCC_CONFIG_CH1_FALLING**, **CTIMER_SELCC_CONFIG_CH2_RISING**, **CTIMER_SELCC_CONFIG_CH2_FALLING**, **CTIMER_SELCC_CONFIG_CH3_RISING**, **CTIMER_SELCC_CONFIG_CH3_FALLING** }
- enum **CTIMER_match_sel_en** { **CTIMER_MATCH_SEL_0** = 0, **CTIMER_MATCH_SEL_1**, **CTIMER_MATCH_SEL_2**, **CTIMER_MATCH_SEL_3** }
- enum **CTIMER_capture_sel_en** { **CTIMER_CAPTURE_SEL_0** = 0, **CTIMER_CAPTURE_SEL_1**, **CTIMER_CAPTURE_SEL_2**, **CTIMER_CAPTURE_SEL_3** }
- enum **CTIMER_external_match_action_en** { **CTIMER_EXTERNAL_MATCH_DO_NOTHING** = 0, **CTIMER_EXTERNAL_MATCH_CLEAR**, **CTIMER_EXTERNAL_MATCH_SET**, **CTIMER_EXTERNAL_MATCH_TOGGLE** }
- enum **CTIMER_mode_en** { **CTIMER_MODE_TIMER** = 0, **CTIMER_MODE_COUNTER_RISING_EDGE**, **CTIMER_MODE_COUNTER_FALLING_EDGE**, **CTIMER_MODE_COUNTER_DUAL_EDGE** }
- enum **CTIMER_count_in_en** { **CTIMER_COUNT_IN_CAP0** = 0, **CTIMER_COUNT_IN_CAP1**, **CTIMER_COUNT_IN_CAP2** }
- enum **CTIMER_capture_reset_edge_en** { **CTIMER_CAPTURE_RESET_CH0_RISING_EDGE** = 0, **CTIMER_CAPTURE_RESET_CH0_FALLING_EDGE**, **CTIMER_CAPTURE_RESET_CH1_RISING_EDGE**, **CTIMER_CAPTURE_RESET_CH1_FALLING_EDGE**, **CTIMER_CAPTURE_RESET_CH2_RISING_EDGE**, **CTIMER_CAPTURE_RESET_CH2_FALLING_EDGE**, **CTIMER_CAPTURE_RESET_CH3_RISING_EDGE**, **CTIMER_CAPTURE_RESET_CH3_FALLING_EDGE** }
- enum **CTIMER_pwm_channel_en** { **CTIMER_PWM_CHANNEL_0** = 0, **CTIMER_PWM_CHANNEL_1**, **CTIMER_PWM_CHANNEL_2** }

Funciones

- static uint8_t CTIMER_get_match_irq_flag (CTIMER_match_sel_en match)

  Obtener estado de flag de interrupcion de algun match.
- static uint8_t CTIMER_get_capture_irq_flag (CTIMER_capture_sel_en capture)

  Obtener estado de flag de interrupcion de algun capture.
- static void CTIMER_clear_match_irq_flag (CTIMER_match_sel_en match)

  Limpiar flag de interrupcion de match.
- static void CTIMER_clear_capture_irq_flag (CTIMER_capture_sel_en capture)

  Limpiar flag de interrupcion de capture.
- static void CTIMER_enable_counter (void)

  Habilitar el contador.
- static void CTIMER_disable_counter (void)

  Inhabilitar el contador.
- static void CTIMER_assert_counter_reset (void)

  Accionar el reset del contador.
- static void CTIMER_clear_counter_reset (void)

  Limpiar el reset del contador.
- static void CTIMER_write_counter (uint32_t value)

  Escribir conteo.
- static uint32_t CTIMER_read_counter (void)

  Leer valor de conteo actual.
- static void CTIMER_write_prescaler (uint32_t value)

  Escribir valor de prescaler.
- static uint32_t CTIMER_read_prescaler (void)

  Leer el valor del prescaler.
- static void CTIMER_enable_interrupt_on_match (CTIMER_match_sel_en match)

  Habilitar interrupcion en match.
- static void CTIMER_disable_interrupt_on_match (CTIMER_match_sel_en match)

  Inhabilitar interrupcion en match.
- static void CTIMER_enable_reset_on_match (CTIMER_match_sel_en match)

  Habilitar reset en match.
- static void CTIMER_disable_reset_on_match (CTIMER_match_sel_en match)

  Inhabilitar reset en match.
- static void CTIMER_enable_stop_on_match (CTIMER_match_sel_en match)

  Habilitar stop en match.
- static void CTIMER_disable_stop_on_match (CTIMER_match_sel_en match)

  Inhabilitar stop en match.
- static void CTIMER_enable_reload_on_match (CTIMER_match_sel_en match)

  Habilitar reload en match.
- static void CTIMER_disable_reload_on_match (CTIMER_match_sel_en match)

  Inhabilitar reload en match.
- static void CTIMER_write_match_value (CTIMER_match_sel_en match, uint32_t value)

  Escribir un registro de match.
- static uint32_t CTIMER_read_match_value (CTIMER_match_sel_en match)

  Leer un registro de match.
- static void CTIMER_enable_rising_edge_capture (CTIMER_capture_sel_en capture)

  Habilitar captura en flanco ascendente.
- static void CTIMER_disable_rising_edge_capture (CTIMER_capture_sel_en capture)

  Inhabilitar captura en flanco ascendente.
- static void CTIMER_enable_falling_edge_capture (CTIMER_capture_sel_en capture)

Habilitar captura en flanco descendente.

- static void CTIMER_disable_falling_edge_capture (CTIMER_capture_sel_en capture)
  Inhabilitar captura en flanco descendente.
- static void CTIMER_enable_interrupt_on_capture (CTIMER_capture_sel_en capture)
  Habilitar interrupcion en captura.
- static void CTIMER_disable_interrupt_on_capture (CTIMER_capture_sel_en capture)
  Inhabilitar interrupcion en captura.
- static uint32_t CTIMER_read_capture_value (CTIMER_capture_sel_en capture)
  Leer registro de captura.
- static uint8_t CTIMER_read_match_status (CTIMER_match_sel_en match)
  Leer estado de match externo.
- static void CTIMER_config_external_match (CTIMER_match_sel_en match, CTIMER_external_match_↩ action_en action)
  Configurar accion en match externo.
- static void CTIMER_config_counter_timer_mode (CTIMER_mode_en mode)
  Configurar modo de funcionamiento.
- static void CTIMER_config_counter_input (CTIMER_count_in_en count_in)
  Configurar entrada de conteo (modo counter rising/falling/dual edge)
- static void CTIMER_enable_count_reset_on_capture (void)
  Habilitar reset de conteo y prescaler en captura.
- static void CTIMER_disable_count_reset_on_capture (void)
  Inhabilitar reset de conteo y prescaler en captura.
- static void CTIMER_config_capture_reset (CTIMER_capture_reset_edge_en capture_sel)
  Configurar captura que genera reset de conteo y prescaler (si esta habilitado)
- static void CTIMER_enable_pwm (CTIMER_pwm_channel_en pwm)
  Habilitar canal de PWM.
- static void CTIMER_disable_pwm (CTIMER_pwm_channel_en pwm)
  Inhabilitar canal de PWM.
- static void CTIMER_write_shadow_register (CTIMER_match_sel_en match, uint32_t value)
  Escribir registros fantasma de match.

Variables

- volatile CTIMER_per_t ∗const CTIMER
  Periferico CTIMER.

7.13.1. Descripción detallada

Definiciones a nivel de abstraccion del periferico CTIMER (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.13.2. Documentación de las estructuras de datos

7.13.2.1. struct CTIMER_CTCR_config_t

Campos de datos

CTIMER_CTMODE_config_en	CTMODE	
CTIMER_CINSEL_config_en	CINSEL	
uint8_t	ENCC	
CTIMER_SELCC_config_en	SELCC	

7.13.2.2. struct CTIMER_EMR_config_t

Campos de datos

CTIMER_EMC_config_en	EMC	
uint8_t	mat_enable	
uint8_t	mat_port	
uint8_t	mat_pin	

7.13.3. Documentación de las funciones

7.13.3.1. CTIMER_get_match_irq_flag()

```
static uint8_t CTIMER_get_match_irq_flag (
             CTIMER_match_sel_en match )  [inline], [static]
```

Obtener estado de flag de interrupcion de algun match.

Parámetros

in	*match*	Numero de match a consultar

Devuelve

Valor del flag actual

7.13.3.2. CTIMER_get_capture_irq_flag()

```
static uint8_t CTIMER_get_capture_irq_flag (
             CTIMER_capture_sel_en capture )  [inline], [static]
```

Obtener estado de flag de interrupcion de algun capture.

Parámetros

in	*match*	Numero de capture a consultar

Devuelve

Valor del flag actual

7.13.3.3. CTIMER_clear_match_irq_flag()

```
static void CTIMER_clear_match_irq_flag (
            CTIMER_match_sel_en match )  [inline], [static]
```

Limpiar flag de interrupcion de match.

Parámetros

in	*match*	Numero de match a limpiar

7.13.3.4. CTIMER_clear_capture_irq_flag()

```
static void CTIMER_clear_capture_irq_flag (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Limpiar flag de interrupcion de capture.

Parámetros

in	*match*	Numero de capture a limpiar

7.13.3.5. CTIMER_write_counter()

```
static void CTIMER_write_counter (
            uint32_t value )  [inline], [static]
```

Escribir conteo.

Parámetros

in	*value*	Valor deseado

7.13.3.6. CTIMER_read_counter()

```
static uint32_t CTIMER_read_counter (
```

```
            void  )  [inline], [static]
```

Leer valor de conteo actual.

Devuelve

Valor actual de conteo

7.13.3.7. CTIMER_write_prescaler()

```
static void CTIMER_write_prescaler (
            uint32_t value )  [inline], [static]
```

Escribir valor de prescaler.

Parámetros

in	*value*	Valor de prescaler deseado

7.13.3.8. CTIMER_read_prescaler()

```
static uint32_t CTIMER_read_prescaler (
            void  )  [inline], [static]
```

Leer el valor del prescaler.

Parámetros

in	*Valor*	de prescaler actual

7.13.3.9. CTIMER_enable_interrupt_on_match()

```
static void CTIMER_enable_interrupt_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Habilitar interrupcion en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.10. CTIMER_disable_interrupt_on_match()

```
static void CTIMER_disable_interrupt_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Inhabilitar interrupcion en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.11. CTIMER_enable_reset_on_match()

```
static void CTIMER_enable_reset_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Habilitar reset en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.12. CTIMER_disable_reset_on_match()

```
static void CTIMER_disable_reset_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Inhabilitar reset en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.13. CTIMER_enable_stop_on_match()

```
static void CTIMER_enable_stop_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Habilitar stop en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.14. CTIMER_disable_stop_on_match()

```
static void CTIMER_disable_stop_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Inhabilitar stop en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.15. CTIMER_enable_reload_on_match()

```
static void CTIMER_enable_reload_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Habilitar reload en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.16. CTIMER_disable_reload_on_match()

```
static void CTIMER_disable_reload_on_match (
            CTIMER_match_sel_en match )  [inline], [static]
```

Inhabilitar reload en match.

Parámetros

in	*match*	Numero de match a configurar

7.13.3.17. CTIMER_write_match_value()

```
static void CTIMER_write_match_value (
            CTIMER_match_sel_en match,
            uint32_t value )  [inline], [static]
```

Escribir un registro de match.

Parámetros

in	*match*	Numero de match a configurar
in	*value*	Valor de match deseado

7.13.3.18. CTIMER_read_match_value()

```
static uint32_t CTIMER_read_match_value (
            CTIMER_match_sel_en match )  [inline], [static]
```

Leer un registro de match.

Parámetros

in	*match*	Numero de match a configurar

Devuelve

Valor de match actual

7.13.3.19. CTIMER_enable_rising_edge_capture()

```
static void CTIMER_enable_rising_edge_capture (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Habilitar captura en flanco ascendente.

Parámetros

in	*capture*	Numero de capture a configurar

7.13.3.20. CTIMER_disable_rising_edge_capture()

```
static void CTIMER_disable_rising_edge_capture (
```

```
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Inhabilitar captura en flanco ascendente.

Parámetros

in	*capture*	Numero de capture a configurar

7.13.3.21. CTIMER_enable_falling_edge_capture()

```
static void CTIMER_enable_falling_edge_capture (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Habilitar captura en flanco descendente.

Parámetros

in	*capture*	Numero de capture a configurar

7.13.3.22. CTIMER_disable_falling_edge_capture()

```
static void CTIMER_disable_falling_edge_capture (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Inhabilitar captura en flanco descendente.

Parámetros

in	*capture*	Numero de capture a configurar

7.13.3.23. CTIMER_enable_interrupt_on_capture()

```
static void CTIMER_enable_interrupt_on_capture (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Habilitar interrupcion en captura.

Parámetros

in	*capture*	Numero de capture a configurar

7.13.3.24. CTIMER_disable_interrupt_on_capture()

```
static void CTIMER_disable_interrupt_on_capture (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Inhabilitar interrupcion en captura.

Parámetros

in	*capture*	Numero de capture a configurar

7.13.3.25. CTIMER_read_capture_value()

```
static uint32_t CTIMER_read_capture_value (
            CTIMER_capture_sel_en capture )  [inline], [static]
```

Leer registro de captura.

Parámetros

in	*capture*	Numero de captura a leer

Devuelve

Valor actual de la captura

7.13.3.26. CTIMER_read_match_status()

```
static uint8_t CTIMER_read_match_status (
            CTIMER_match_sel_en match )  [inline], [static]
```

Leer estado de match externo.

Parámetros

in	*match*	Numero de match externo a consultar

Devuelve

Estado del match actual

7.13.3.27. CTIMER_config_external_match()

```
static void CTIMER_config_external_match (
            CTIMER_match_sel_en match,
            CTIMER_external_match_action_en action )  [inline], [static]
```

Configurar accion en match externo.

Parámetros

in	*match*	Numero de match a configurar
in	*action*	Accion a realizar en match externo

7.13.3.28. CTIMER_config_counter_timer_mode()

```
static void CTIMER_config_counter_timer_mode (
            CTIMER_mode_en mode )  [inline], [static]
```

Configurar modo de funcionamiento.

Parámetros

in	*mode*	Modo de funcionamiento

7.13.3.29. CTIMER_config_counter_input()

```
static void CTIMER_config_counter_input (
            CTIMER_count_in_en count_in )  [inline], [static]
```

Configurar entrada de conteo (modo counter rising/falling/dual edge)

Parámetros

in	*count_in*	Entrada de conteo deseada

7.13.3.30. CTIMER_config_capture_reset()

```
static void CTIMER_config_capture_reset (
            CTIMER_capture_reset_edge_en capture_sel )  [inline], [static]
```

Configurar captura que genera reset de conteo y prescaler (si esta habilitado)

Parámetros

in	*capture_sel*	Captura que genera el reset de conteo y prescaler

7.13.3.31. CTIMER_enable_pwm()

```
static void CTIMER_enable_pwm (
            CTIMER_pwm_channel_en pwm )  [inline], [static]
```

Habilitar canal de PWM.

Parámetros

in	*pwm*	Canal deseado

7.13.3.32. CTIMER_disable_pwm()

```
static void CTIMER_disable_pwm (
            CTIMER_pwm_channel_en pwm )  [inline], [static]
```

Inhabilitar canal de PWM.

Parámetros

in	*pwm*	Canal deseado

7.13.3.33. CTIMER_write_shadow_register()

```
static void CTIMER_write_shadow_register (
            CTIMER_match_sel_en match,
            uint32_t value )  [inline], [static]
```

Escribir registros fantasma de match.

Parámetros

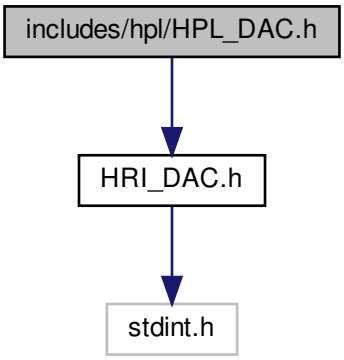
in	*match*	Match a escribir
in	*value*	Valor deseado

7.14. Referencia del Archivo includes/hpl/HPL_DAC.h

Declaraciones a nivel de abstraccion de periferico del DAC (LPC845)

```
#include <HRI_DAC.h>
```

Dependencia gráfica adjunta para HPL_DAC.h:



Enumeraciones

- enum **DAC_sel_en** { **DAC_SEL_0** = 0, **DAC_SEL_1** }
- enum **DAC_settling_time_en** { **DAC_SETTLING_TIME_SEL_1US_MAX** = 0, **DAC_SETTLING_TIME_↩ SEL_2_5US_MAX** }

Funciones

- static void DAC_write (DAC_sel_en dac, uint16_t new_value)
  Actualizacion del valor actual del DAC.
- static void DAC_config_settling_time (DAC_sel_en dac, DAC_settling_time_en settling_time)
  Configuracion del settling time del DAC.
- static void DAC_enable_DMA_request (DAC_sel_en dac)
  Habilitar interrupcion de DMA cuando el timer tiemoutea.
- static void DAC_disable_DMA_request (DAC_sel_en dac)
  Inhabilitar interrupcion de DMA cuando el timer tiemoutea.
- static void DAC_enable_double_buffer (DAC_sel_en dac)
  Habilitar double buffering.
- static void DAC_disable_double_buffer (DAC_sel_en dac)
  Inhabilitar double buffering.
- static void DAC_enable_timer (DAC_sel_en dac)
  Habilitar operacion del timer.
- static void DAC_disable_timer (DAC_sel_en dac)
  Inhabilitar operacion del timer.
- static void DAC_enable_DMA (DAC_sel_en dac)
  Habilitar DMA request asociada al DAC.
- static void DAC_disable_DMA (DAC_sel_en dac)
  Inhabilitar DMA request asociada al DAC.
- static void DAC_write_reaload_value (DAC_sel_en dac, uint16_t value)
  Escribir valor a recargar para el timer de DMA.

Variables

- volatile DAC_per_t *const DAC [ ]

  Perifericos DAC.

7.14.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del DAC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.14.2. Documentación de las funciones

7.14.2.1. DAC_write()

```
static void DAC_write (
            DAC_sel_en dac,
            uint16_t new_value )  [inline], [static]
```

Actualizacion del valor actual del DAC.

Parámetros

in	*dac*	Instancia a actualizar
in	*new_value*	Nuevo valor a poner en el DAC

7.14.2.2. DAC_config_settling_time()

```
static void DAC_config_settling_time (
            DAC_sel_en dac,
            DAC_settling_time_en settling_time )  [inline], [static]
```

Configuracion del settling time del DAC.

Parámetros

in	*dac*	Instancia a configurar
in	*settling_time*	Configuracion deseada

7.14.2.3. DAC_enable_DMA_request()

```
static void DAC_enable_DMA_request (
            DAC_sel_en dac )  [inline], [static]
```

Habilitar interrupcion de DMA cuando el timer tiemoutea.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.4. DAC_disable_DMA_request()

```
static void DAC_disable_DMA_request (
            DAC_sel_en dac )  [inline], [static]
```

Inhabilitar interrupcion de DMA cuando el timer tiemoutea.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.5. DAC_enable_double_buffer()

```
static void DAC_enable_double_buffer (
            DAC_sel_en dac )  [inline], [static]
```

Habilitar double buffering.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.6. DAC_disable_double_buffer()

```
static void DAC_disable_double_buffer (
            DAC_sel_en dac )  [inline], [static]
```

Inhabilitar double buffering.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.7. DAC_enable_timer()

```
static void DAC_enable_timer (
            DAC_sel_en dac )  [inline], [static]
```

Habilitar operacion del timer.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.8. DAC_disable_timer()

```
static void DAC_disable_timer (
            DAC_sel_en dac )  [inline], [static]
```

Inhabilitar operacion del timer.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.9. DAC_enable_DMA()

```
static void DAC_enable_DMA (
            DAC_sel_en dac )  [inline], [static]
```

Habilitar DMA request asociada al DAC.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.10. DAC_disable_DMA()

```
static void DAC_disable_DMA (
            DAC_sel_en dac )  [inline], [static]
```

Inhabilitar DMA request asociada al DAC.

Parámetros

in	*dac*	Instancia a configurar

7.14.2.11. DAC_write_reaload_value()

```
static void DAC_write_reaload_value (
            DAC_sel_en dac,
            uint16_t value )  [inline], [static]
```

Escribir valor a recargar para el timer de DMA.

Parámetros

in	*dac*	Instancia a configurar
in	*value*	Valor deseado

7.15. Referencia del Archivo includes/hpl/HPL_GPIO.h

Declaraciones a nivel de abstraccion de periferico del GPIO (LPC845)

```
#include <HRI_GPIO.h>
```

Dependencia gráfica adjunta para HPL_GPIO.h:

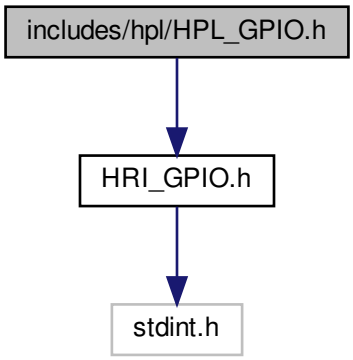
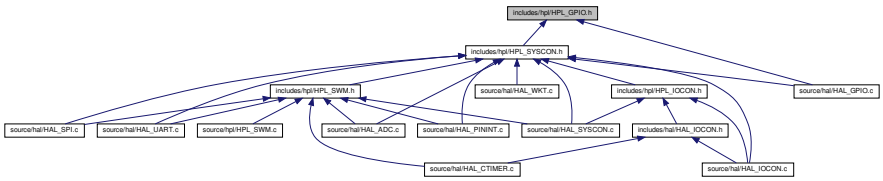


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **GPIO_dir_en** { **GPIO_DIR_INPUT** = 0, **GPIO_DIR_OUTPUT** }
- enum **GPIO_port_en** { **GPIO_PORT_0** = 0, **GPIO_PORT_1** }
- enum **GPIO_portpin_en** {
  GPIO_PORTPIN_0_0 = 0, **GPIO_PORTPIN_0_1**, **GPIO_PORTPIN_0_2**, **GPIO_PORTPIN_0_3**, **GPIO_PORTPIN_0_4**, **GPIO_PORTPIN_0_5**, **GPIO_PORTPIN_0_6**, **GPIO_PORTPIN_0_7**, **GPIO_PORTPIN_0_8**, **GPIO_PORTPIN_0_9**, **GPIO_PORTPIN_0_10**, **GPIO_PORTPIN_0_11**, **GPIO_PORTPIN_0_12**, **GPIO_PORTPIN_0_13**, **GPIO_PORTPIN_0_14**, **GPIO_PORTPIN_0_15**, **GPIO_PORTPIN_0_16**, **GPIO_PORTPIN_0_17**, **GPIO_PORTPIN_0_18**, **GPIO_PORTPIN_0_19**, **GPIO_PORTPIN_0_20**, **GPIO_PORTPIN_0_21**, **GPIO_PORTPIN_0_22**, **GPIO_PORTPIN_0_23**, **GPIO_PORTPIN_0_24**, **GPIO_PORTPIN_0_25**, **GPIO_PORTPIN_0_26**, **GPIO_PORTPIN_0_27**, **GPIO_PORTPIN_0_28**, **GPIO_PORTPIN_0_29**, **GPIO_PORTPIN_0_30**, **GPIO_PORTPIN_0_31**, **GPIO_PORTPIN_1_0**, **GPIO_PORTPIN_1_1**, **GPIO_PORTPIN_1_2**, **GPIO_PORTPIN_1_3**, **GPIO_PORTPIN_1_4**, **GPIO_PORTPIN_1_5**, **GPIO_PORTPIN_1_6**, **GPIO_PORTPIN_1_7**, **GPIO_PORTPIN_1_8**, **GPIO_PORTPIN_1_9**, **GPIO_PORTPIN_1_10**, **GPIO_PORTPIN_1_11**, **GPIO_PORTPIN_1_12**, **GPIO_PORTPIN_1_13**, **GPIO_PORTPIN_1_14**, **GPIO_PORTPIN_1_15**, **GPIO_PORTPIN_1_16**, **GPIO_PORTPIN_1_17**, **GPIO_PORTPIN_1_18**, **GPIO_PORTPIN_1_19**, **GPIO_PORTPIN_1_20**, **GPIO_PORTPIN_1_21** }

Funciones

- static uint8_t GPIO_read_port_byte (GPIO_portpin_en portpin)

  Leer estado del pin absoluto (sin importar mascaras ni funcion alternativa)
- static void GPIO_write_port_byte (GPIO_portpin_en portpin, uint8_t value)

  Escribir estado del pin absoluto (sin importar mascaras ni funcion alternativa)
- static uint8_t GPIO_read_port_word (GPIO_portpin_en portpin)

  Leer estado del pin absoluto (sin importar mascaras ni funcion alternativa)
- static void GPIO_write_port_word (GPIO_portpin_en portpin, uint8_t value)

  Escribir estado del pin absoluto (sin importar mascaras ni funcion alternativa)
- static uint32_t GPIO_read_dir (GPIO_port_en port)

  Leer registro de direccion.
- static void GPIO_write_dir (GPIO_port_en port, uint32_t value)

  Escribir registro de direccion.
- static uint32_t GPIO_read_mask (GPIO_port_en port)

  Leer registro de mascara.
- static void GPIO_write_mask (GPIO_port_en port, uint32_t value)

  Escribir registro de mascara.
- static uint32_t GPIO_read_portpin (GPIO_port_en port)

  Leer registro de puerto/pin.
- static void GPIO_write_portpin (GPIO_port_en port, uint32_t value)

  Escribir registro de puerto/pin.
- static uint32_t GPIO_read_masked_portpin (GPIO_port_en port)

  Leer registro de puerto/pin enmascarado.
- static void GPIO_write_masked_portpin (GPIO_port_en port, uint32_t value)

  Escribir registro de puerto/pin enmascarado.
- static void GPIO_write_set (GPIO_port_en port, uint32_t value)

  Escribir registro de set.
- static void GPIO_write_clear (GPIO_port_en port, uint32_t value)

  Escribir registro de clear.
- static void GPIO_write_toggle (GPIO_port_en port, uint32_t value)

  Escribir registro de toggle.
- static void GPIO_write_dir_set (GPIO_port_en port, uint32_t value)

  Escribir registro de direction set.
- static void GPIO_write_dir_clear (GPIO_port_en port, uint32_t value)

  Escribir registro de direction clear.
- static void GPIO_write_dir_toggle (GPIO_port_en port, uint32_t value)

  Escribir registro de direction toggle.

Variables

- volatile GPIO_per_t ∗const GPIO

  Periferico GPIO.

7.15.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del GPIO (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.15.2. Documentación de las funciones

7.15.2.1. GPIO_read_port_byte()

```
static uint8_t GPIO_read_port_byte (
            GPIO_portpin_en portpin )  [inline], [static]
```

Leer estado del pin absoluto (sin importar mascaras ni funcion alternativa)

Parámetros

in	*portpin*	Numero de port/pin a consultar

Devuelve

Estado del pin absoluto

7.15.2.2. GPIO_write_port_byte()

```
static void GPIO_write_port_byte (
            GPIO_portpin_en portpin,
            uint8_t value )  [inline], [static]
```

Escribir estado del pin absoluto (sin importar mascaras ni funcion alternativa)

Parámetros

in	*portpin*	Numero de port/pin a escribir
in	*value*	Valor a escribir

7.15.2.3. GPIO_read_port_word()

```
static uint8_t GPIO_read_port_word (
            GPIO_portpin_en portpin )  [inline], [static]
```

Leer estado del pin absoluto (sin importar mascaras ni funcion alternativa)

Parámetros

in	*portpin*	Numero de port/pin a consultar

Devuelve

Estado del pin absoluto

7.15.2.4. GPIO_write_port_word()

```
static void GPIO_write_port_word (
            GPIO_portpin_en portpin,
            uint8_t value )  [inline], [static]
```

Escribir estado del pin absoluto (sin importar mascaras ni funcion alternativa)

Parámetros

in	*portpin*	Numero de port/pin a escribir
in	*value*	Valor a escribir

7.15.2.5. GPIO_read_dir()

```
static uint32_t GPIO_read_dir (
            GPIO_port_en port )  [inline], [static]
```

Leer registro de direccion.

Parámetros

in	*port*	Numero de puerto a consultar

Devuelve

Valor del registro

7.15.2.6. GPIO_write_dir()

```
static void GPIO_write_dir (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de direccion.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.7. GPIO_read_mask()

```
static uint32_t GPIO_read_mask (
            GPIO_port_en port )  [inline], [static]
```

Leer registro de mascara.

Parámetros

in	*port*	Numero de puerto a consultar

Devuelve

Valor del registro

7.15.2.8. GPIO_write_mask()

```
static void GPIO_write_mask (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de mascara.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.9. GPIO_read_portpin()

```
static uint32_t GPIO_read_portpin (
            GPIO_port_en port )  [inline], [static]
```

Leer registro de puerto/pin.

Parámetros

in	*port*	Numero de puerto a consultar

Devuelve

Valor del registro

7.15.2.10. GPIO_write_portpin()

```
static void GPIO_write_portpin (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de puerto/pin.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.11. GPIO_read_masked_portpin()

```
static uint32_t GPIO_read_masked_portpin (
            GPIO_port_en port )  [inline], [static]
```

Leer registro de puerto/pin enmascarado.

Parámetros

in	*port*	Numero de puerto a consultar

Devuelve

Valor del registro

7.15.2.12. GPIO_write_masked_portpin()

```
static void GPIO_write_masked_portpin (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de puerto/pin enmascarado.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.13. GPIO_write_set()

```
static void GPIO_write_set (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de set.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.14. GPIO_write_clear()

```
static void GPIO_write_clear (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de clear.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.15. GPIO_write_toggle()

```
static void GPIO_write_toggle (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de toggle.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.16. GPIO_write_dir_set()

```
static void GPIO_write_dir_set (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de direction set.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.17. GPIO_write_dir_clear()

```
static void GPIO_write_dir_clear (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de direction clear.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.15.2.18. GPIO_write_dir_toggle()

```
static void GPIO_write_dir_toggle (
            GPIO_port_en port,
            uint32_t value )  [inline], [static]
```

Escribir registro de direction toggle.

Parámetros

in	*port*	Numero de puerto a configurar
in	*value*	Valor deseado

7.16. Referencia del Archivo includes/hpl/HPL_IOCON.h

Declaraciones a nivel de abstraccion de periferico del IOCON (LPC845)

```
#include <HPL_SYSCON.h>
#include <HRI_IOCON.h>
```

Dependencia gráfica adjunta para HPL_IOCON.h:

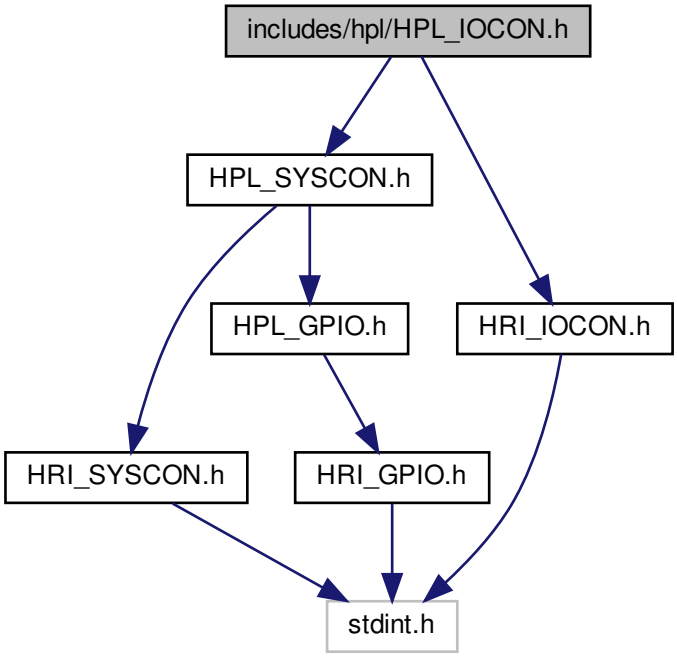
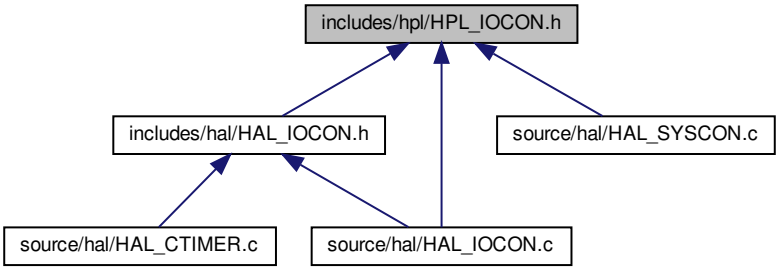


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **IOCON_pull_mode_en** { **IOCON_PULL_NONE** = 0, **IOCON_PULL_DOWN**, **IOCON_PULL_UP**, **IOCON_PULL_REPEATER** }
- enum **IOCON_sample_mode_en** { **IOCON_SAMPLE_MODE_BYPASS** = 0, **IOCON_SAMPLE_MODE_1_CLOCK**, **IOCON_SAMPLE_MODE_2_CLOCK**, **IOCON_SAMPLE_MODE_3_CLOCK** }
- enum **IOCON_clk_sel_en** { **IOCON_CLK_DIV_0** = 0, **IOCON_CLK_DIV_1**, **IOCON_CLK_DIV_2**, **IOCON_CLK_DIV_3**, **IOCON_CLK_DIV_4**, **IOCON_CLK_DIV_5**, **IOCON_CLK_DIV_6** }
- enum **IOCON_iic_mode_en** { **IOCON_IIC_MODE_STANDARD** = 0, **IOCON_IIC_MODE_GPIO**, **IOCON_IIC_MODE_FAST_MODE** }

Funciones

- static void IOCON_init (void)

  Inicializacion del modulo IOCON.
- static void IOCON_deinit (void)

  Inhabilitacion del modulo IOCON.
- static void IOCON_config_pull_mode (uint8_t port, uint8_t pin, IOCON_pull_mode_en pull_mode)

  Configurar modo de funcionamiento (pull up, pull down, etc) en un pin.
- static void IOCON_enable_hysteresis (uint8_t port, uint8_t pin)

  Habilitar histeresis en un pin.
- static void IOCON_disable_hysteresis (uint8_t port, uint8_t pin)

  Inhabilitar histeresis en un pin.
- static void IOCON_enable_invert (uint8_t port, uint8_t pin)

  Habilitar inversion en un pin.
- static void IOCON_disable_invert (uint8_t port, uint8_t pin)

  Inhabilitar inversion en un pin.
- static void IOCON_enable_open_drain (uint8_t port, uint8_t pin)

  Habilitar modo open drain en un pin.
- static void IOCON_disable_open_drain (uint8_t port, uint8_t pin)

  Inhabilitar modo open drain en un pin.
- static void IOCON_config_sample_mode (uint8_t port, uint8_t pin, IOCON_sample_mode_en sample_mode)

  Configurar modo de sampleo en un pin.
- static void IOCON_config_clock_source (uint8_t port, uint8_t pin, IOCON_clk_sel_en clock_source)

Configurar fuente utilizada para el sampleo en un pin.

- static void IOCON_enable_dac0 (void)

  Habilitar DAC0 en PIO0_17.
- static void IOCON_enable_dac1 (void)

  Habilitar DAC1 en PIO0_29.
- static void IOCON_disable_dac0 (void)

  Inhabilitar DAC0 en PIO0_17.
- static void IOCON_disable_dac1 (void)

  Inhabilitar DAC1 en PIO0_29.
- static void IOCON_select_iic0_scl (IOCON_iic_mode_en iic_mode)

  Habilitar IIC0_SCL en PIO0_10.
- static void IOCON_select_iic0_sda (IOCON_iic_mode_en iic_mode)

  Habilitar IIC SDA en PIO0_11.

Variables

- volatile IOCON_per_t ∗const IOCON

  Periferico IOCON.
- volatile IOCON_PIO_reg_t ∗const IOCON_PIN_TABLE [2][32]

  Tabla de registros de configuracion.

7.16.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del IOCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.16.2. Documentación de las funciones

7.16.2.1. IOCON_init()

```
static void IOCON_init (
            void  )  [inline], [static]
```

Inicializacion del modulo IOCON.

Unicamente habilita el clock del modulo

7.16.2.2. IOCON_deinit()

```
static void IOCON_deinit (
            void  )  [inline], [static]
```

Inhabilitacion del modulo IOCON.

Unicamente inhabilita el clock del modulo

7.16.2.3. IOCON_config_pull_mode()

```
static void IOCON_config_pull_mode (
            uint8_t port,
            uint8_t pin,
            IOCON_pull_mode_en pull_mode )  [inline], [static]
```

Configurar modo de funcionamiento (pull up, pull down, etc) en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin
in	*pull_mode*	Modo de funcionamiento

7.16.2.4. IOCON_enable_hysteresis()

```
static void IOCON_enable_hysteresis (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Habilitar histeresis en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin

7.16.2.5. IOCON_disable_hysteresis()

```
static void IOCON_disable_hysteresis (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Inhabilitar histeresis en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin

7.16.2.6. IOCON_enable_invert()

```
static void IOCON_enable_invert (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Habilitar inversion en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin

7.16.2.7. IOCON_disable_invert()

```
static void IOCON_disable_invert (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Inhabilitar inversion en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin

7.16.2.8. IOCON_enable_open_drain()

```
static void IOCON_enable_open_drain (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Habilitar modo open drain en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin

7.16.2.9. IOCON_disable_open_drain()

```
static void IOCON_disable_open_drain (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Inhabilitar modo open drain en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin

7.16.2.10. IOCON_config_sample_mode()

```
static void IOCON_config_sample_mode (
            uint8_t port,
            uint8_t pin,
            IOCON_sample_mode_en sample_mode )  [inline], [static]
```

Configurar modo de sampleo en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin
in	*sample_mode*	Modo de sampleo

7.16.2.11. IOCON_config_clock_source()

```
static void IOCON_config_clock_source (
            uint8_t port,
            uint8_t pin,
            IOCON_clk_sel_en clock_source )  [inline], [static]
```

Configurar fuente utilizada para el sampleo en un pin.

Parámetros

in	*port*	Numero de puerto
in	*pin*	Numero de pin
in	*clock_source*	Fuente de clock deseada

7.16.2.12. IOCON_select_iic0_scl()

```
static void IOCON_select_iic0_scl (
            IOCON_iic_mode_en iic_mode )  [inline], [static]
```

Habilitar IIC0_SCL en PIO0_10.

Parámetros

in	*iic_mode*	Modo de IIC

7.16.2.13. IOCON_select_iic0_sda()

```
static void IOCON_select_iic0_sda (
            IOCON_iic_mode_en iic_mode )  [inline], [static]
```

Habilitar IIC SDA en PIO0_11.

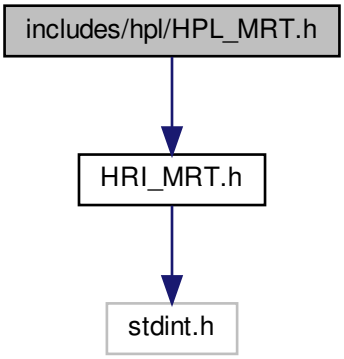
Parámetros

in	*iic_mode*	Modo de IIC

7.17. Referencia del Archivo includes/hpl/HPL_MRT.h

Declaraciones a nivel de abstraccion de periferico del MRT (LPC845)

`#include <HRI_MRT.h>`
Dependencia gráfica adjunta para HPL_MRT.h:



Enumeraciones

- enum **MRT_channel_sel_en** { **MRT_CHANNEL_0** = 0, **MRT_CHANNEL_1**, **MRT_CHANNEL_2**, **MRT_↩ CHANNEL_3** }
- enum **MRT_mode_en** { **MRT_MODE_REPEAT** = 0, **MRT_MODE_ONE_SHOT**, **MRT_MODE_ONE_SHO↩ T_BUS_STALL** }

Funciones

- static void MRT_set_interval (MRT_channel_sel_en channel, uint32_t interval)

  Fijar intervalo de un canal del MRT sin detener el conteo actual.
- static void MRT_set_interval_and_stop_timer (MRT_channel_sel_en channel, uint32_t interval)

  Fijar intervalo de un canal del MRT deteniendo el conteo actual inmediatamente.
- static uint32_t MRT_get_current_value (MRT_channel_sel_en channel)

  Obtener el valor de la cuenta actual de un canal del MRT.
- static void **MRT_enable_irq** (MRT_channel_sel_en channel)
- static void **MRT_disable_irq** (MRT_channel_sel_en channel)
- static void MRT_config_mode (MRT_channel_sel_en channel, MRT_mode_en mode)

  Configurar modo de funcionamiento de un canal del MRT.
- static uint8_t MRT_get_idle_channel (void)

  Obtener el canal que este en estado IDLE.
- static uint8_t MRT_get_irq_flag (MRT_channel_sel_en channel)

  Obtener flag de interrupcion de un canal.
- static void MRT_clear_irq_flag (MRT_channel_sel_en channel)

  Limpiar flag de interrupcion de un canal.

Variables

- volatile MRT_per_t ∗const MRT

  Periferico MRT.

7.17.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del MRT (LPC845)

Autor

Augusto Santini

Fecha

4/2020

Versión

1.0

7.17.2. Documentación de las funciones

7.17.2.1. MRT_set_interval()

```
static void MRT_set_interval (
            MRT_channel_sel_en channel,
            uint32_t interval )  [inline], [static]
```

Fijar intervalo de un canal del MRT sin detener el conteo actual.

Parámetros

in	*channel*	Canal a configurar
in	*interval*	Intervalo a cargar

7.17.2.2. MRT_set_interval_and_stop_timer()

```
static void MRT_set_interval_and_stop_timer (
            MRT_channel_sel_en channel,
            uint32_t interval )  [inline], [static]
```

Fijar intervalo de un canal del MRT deteniendo el conteo actual inmediatamente.

Parámetros

in	*channel*	Canal a configurar
in	*interval*	Intervalo a cargar

7.17.2.3. MRT_get_current_value()

```
static uint32_t MRT_get_current_value (
            MRT_channel_sel_en channel )  [inline], [static]
```

Obtener el valor de la cuenta actual de un canal del MRT.

Parámetros

in	*channel*	Canal a consultar

Devuelve

Cuenta actual

7.17.2.4. MRT_config_mode()

```
static void MRT_config_mode (
            MRT_channel_sel_en channel,
            MRT_mode_en mode )  [inline], [static]
```

Configurar modo de funcionamiento de un canal del MRT.

Parámetros

in	*channel*	Canal a configurar
in	*mode*	Modo deseado

7.17.2.5. MRT_get_idle_channel()

```
static uint8_t MRT_get_idle_channel (
            void  )  [inline], [static]
```

Obtener el canal que este en estado IDLE.

Devuelve

Menor canal de los que esten en estado IDLE

7.17.2.6. MRT_get_irq_flag()

```
static uint8_t MRT_get_irq_flag (
            MRT_channel_sel_en channel )  [inline], [static]
```

Obtener flag de interrupcion de un canal.

Parámetros

in	*channel*	Canal a consultar

Devuelve

Flag actual de interrupcion del canal consultado

7.17.2.7. MRT_clear_irq_flag()

```
static void MRT_clear_irq_flag (
            MRT_channel_sel_en channel )  [inline], [static]
```

Limpiar flag de interrupcion de un canal.

Parámetros

in	*channel*	Canal a consultar

7.18. Referencia del Archivo includes/hpl/HPL_NVIC.h

Declaraciones a nivel de abstraccion de periferico del NVIC (LPC845)

```
#include <HRI_NVIC.h>
```

Dependencia gráfica adjunta para HPL_NVIC.h:

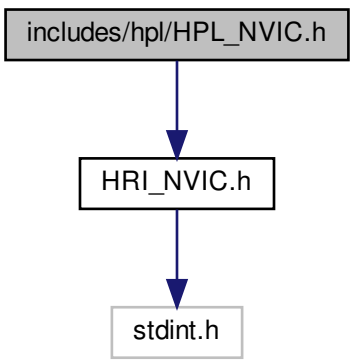
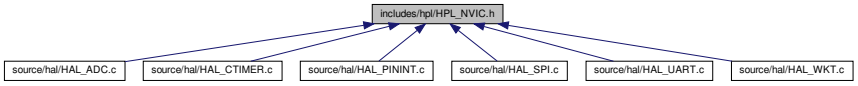


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **NVIC_irq_sel_en** {
  NVIC_IRQ_SEL_SPI0 = 0, **NVIC_IRQ_SEL_SPI1**, **NVIC_IRQ_SEL_DAC0**, **NVIC_IRQ_SEL_UART0**,
  NVIC_IRQ_SEL_UART1, **NVIC_IRQ_SEL_UART2**, **NVIC_IRQ_SEL_IIC1** = 7, **NVIC_IRQ_SEL_IIC0**,
  NVIC_IRQ_SEL_SCT, **NVIC_IRQ_SEL_MRT**, **NVIC_IRQ_SEL_CMP_CAPT**, **NVIC_IRQ_SEL_WDT**,
  NVIC_IRQ_SEL_BOD, **NVIC_IRQ_SEL_FLASH**, **NVIC_IRQ_SEL_WKT**, **NVIC_IRQ_SEL_ADC_SEQA**,
  NVIC_IRQ_SEL_ADC_SEQB, **NVIC_IRQ_SEL_ADC_THCMP**, **NVIC_IRQ_SEL_ADC_OVR**, **NVIC_IRQ↩_SEL_DMA**,
  NVIC_IRQ_SEL_IIC2, **NVIC_IRQ_SEL_IIC3**, **NVIC_IRQ_SEL_CTIMER**, **NVIC_IRQ_SEL_PININT0**,
  NVIC_IRQ_SEL_PININT1, **NVIC_IRQ_SEL_PININT2**, **NVIC_IRQ_SEL_PININT3**, **NVIC_IRQ_SEL_PININ↩T4**,
  NVIC_IRQ_SEL_PININT5_DAC1, **NVIC_IRQ_SEL_PININT6_UART3**, **NVIC_IRQ_SEL_PININT7_UART4** }
- enum **NVIC_irq_priority_en** { **NVIC_IRQ_PRIORITY_HIGHEST** = 0, **NVIC_IRQ_PRIORITY_HIGH**, **NVI↩C_IRQ_PRIORITY_LOW**, **NVIC_IRQ_PRIORITY_LOWEST** }

Funciones

- static void NVIC_enable_interrupt (NVIC_irq_sel_en irq)

  Habilitacion de interrupciones.
- static void NVIC_disable_interrupt (NVIC_irq_sel_en irq)

  Inhabilitacion de interrupciones.
- static void NVIC_set_pending_interrupt (NVIC_irq_sel_en irq)

  Fijar interupcion pendiente por software.
- static void NVIC_clear_pending_interrupt (NVIC_irq_sel_en irq)

  Limpiar interupcion pendiente por software.
- static uint8_t NVIC_get_active_interrupt (NVIC_irq_sel_en irq)

  Obtener estado de interrupcion.
- static void **NVIC_set_irq_priority** (NVIC_irq_sel_en irq, NVIC_irq_priority_en priority)

Variables

- volatile NVIC_per_t ∗const NVIC

  Periferico NVIC.

7.18.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del NVIC (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.18.2. Documentación de las funciones

7.18.2.1. NVIC_enable_interrupt()

```
static void NVIC_enable_interrupt (
            NVIC_irq_sel_en irq )  [inline], [static]
```

Habilitacion de interrupciones.

Parámetros

in	*irq*	Seleccion de fuente de interrupcion

7.18.2.2. NVIC_disable_interrupt()

```
static void NVIC_disable_interrupt (
            NVIC_irq_sel_en irq )  [inline], [static]
```

Inhabilitacion de interrupciones.

Parámetros

in	*irq*	Seleccion de fuente de interrupcion

7.18.2.3. NVIC_set_pending_interrupt()

```
static void NVIC_set_pending_interrupt (
            NVIC_irq_sel_en irq )  [inline], [static]
```

Fijar interupcion pendiente por software.

Parámetros

in	*irq*	Seleccion de fuente de interrupcion

7.18.2.4. NVIC_clear_pending_interrupt()

```
static void NVIC_clear_pending_interrupt (
            NVIC_irq_sel_en irq )  [inline], [static]
```

Limpiar interupcion pendiente por software.

Parámetros

in	*irq*	Seleccion de fuente de interrupcion

7.18.2.5. NVIC_get_active_interrupt()

```
static uint8_t NVIC_get_active_interrupt (
            NVIC_irq_sel_en irq )  [inline], [static]
```

Obtener estado de interrupcion.

Parámetros

in	*irq*	Seleccion de fuente de interrupcion

Devuelve

Si la interrupcion estaba activa devuelve 1, caso contrario devuelve 0

7.19. Referencia del Archivo includes/hpl/HPL_PININT.h

Declaraciones a nivel de abstraccion de periferico del PININT (LPC845)

```
#include <HRI_PININT.h>
```
Dependencia gráfica adjunta para HPL_PININT.h:

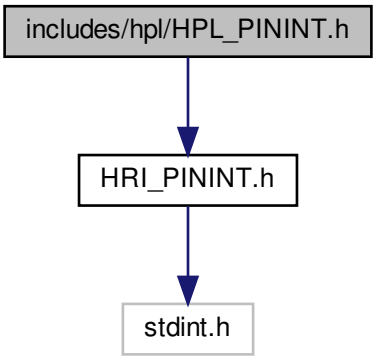
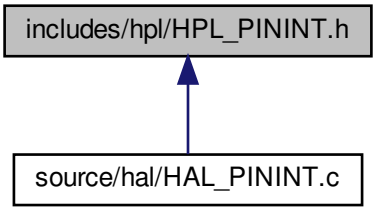


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **PININT_interrupt_mode_en** { **PININT_INTERRUPT_MODE_EDGE** = 0, **PININT_INTERRUPT_M↩ ODE_LEVEL** }
- enum **PININT_match_contribution_en** { **PININT_MATCH_CONTRIBUTION_CONSTANT_HIGH** = 0, **PININT_MATCH_CONTRIBUTION_STICKY↩ _RISING_EDGE**, **PININT_MATCH_CONTRIBUTION_STICKY_FALLING_EDGE**, **PININT_MATCH_CON↩ TRIBUTION_STICKY_RISING_OR_FALLING_EDGE**, **PININT_MATCH_CONTRIBUTION_HIGH_LEVEL**, **PININT_MATCH_CONTRIBUTION_LOW_LEVEL**, **P↩ ININT_MATCH_CONTRIBUTION_CONSTANT_0**, **PININT_MATCH_CONTRIBUTION_EVENT** }

Funciones

- static void PININT_set_interrupt_mode (uint8_t channel, PININT_interrupt_mode_en mode)

  Configurar sensibilidad a nivel/flanco.

- static PININT_interrupt_mode_en PININT_get_interrupt_mode (uint8_t channel)

  Obtener configuracion de modo de un canal.
- static void PININT_enable_rising_edge (uint8_t channel)

  Habilitar detecciones por flanco ascendente.
- static void PININT_disable_rising_edge (uint8_t channel)

  Inhabilitar detecciones por flanco ascendente.
- static void PININT_enable_falling_edge (uint8_t channel)

  Habilitar detecciones por flanco descendente.
- static void PININT_disable_falling_edge (uint8_t channel)

  Inhabilitar detecciones por flanco descendente.
- static void PININT_enable_high_level (uint8_t channel)

  Habilitar detecciones por nivel alto.
- static void PININT_disable_high_level (uint8_t channel)

  Inhabilitar detecciones por flanco ascendente.
- static uint8_t PININT_get_rising_edge_active (void)

  Obtener interrupciones activas por flanco ascendente.
- static uint8_t PININT_get_falling_edge_active (void)

  Obtener interrupciones activas por flanco descendente.
- static uint8_t PININT_get_level_active (void)

  Obtener interrupciones activas por nivel.
- static void PININT_clear_edge_level_irq (uint8_t channel)

  Limpiar flag de interrupcion por flanco.
- static void PININT_toggle_active_level (uint8_t channel)

  Invertir nivel activo de itnerrupcion por nivel.
- static void PININT_enable_pattern_match (void)

  Habilita el funcionamiento del pattern match engine (inhabilita PININT)
- static void PININT_disable_pattern_match (void)

  Inhabilita el funcionamiento del pattern match engine (habilita PININT)
- static void PININT_enable_RXEV (void)

  Habilita la salida de RXEV.
- static void PININT_disable_RXEV (void)

  Inhabilita la salida de RXEV.
- static uint8_t PININT_get_pattern_match_state (void)

  Obtener estado actual del pattern match.
- static void PININT_config_pattern_match_source (uint8_t channel, uint8_t slice)

  Configuracion de las fuentes del pattern match.
- static void PINITN_enable_slice_as_endpoint (uint8_t slice)

  Habilitar slice como endpoint.
- static void PINITN_disable_slice_as_endpoint (uint8_t slice)

  Inhabilitar slice como endpoint.
- static void PINITN_config_slice_mode (uint8_t slice, PININT_match_contribution_en mode)

  Configurar modo del slice.

Variables

- volatile PININT_per_t ∗const PININT

  Periferico PININT.

7.19.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del PININT (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.19.2. Documentación de las funciones

7.19.2.1. PININT_set_interrupt_mode()

```
static void PININT_set_interrupt_mode (
            uint8_t channel,
            PININT_interrupt_mode_en mode )  [inline], [static]
```

Configurar sensibilidad a nivel/flanco.

Parámetros

in	*channel*	Canal a configurar
in	*mode*	Modo deseado

7.19.2.2. PININT_get_interrupt_mode()

```
static PININT_interrupt_mode_en PININT_get_interrupt_mode (
            uint8_t channel )  [inline], [static]
```

Obtener configuracion de modo de un canal.

Parámetros

in	*channel*	Canal a consultar

Devuelve

Modo configurado para el canal

7.19.2.3. PININT_enable_rising_edge()

```
static void PININT_enable_rising_edge (
            uint8_t channel )  [inline], [static]
```

Habilitar detecciones por flanco ascendente.

Parámetros

in	*channel*	Canal deseado

7.19.2.4. PININT_disable_rising_edge()

```
static void PININT_disable_rising_edge (
            uint8_t channel )  [inline], [static]
```

Inhabilitar detecciones por flanco ascendente.

Parámetros

in	*channel*	Canal deseado

7.19.2.5. PININT_enable_falling_edge()

```
static void PININT_enable_falling_edge (
            uint8_t channel )  [inline], [static]
```

Habilitar detecciones por flanco descendente.

Parámetros

in	*channel*	Canal deseado

7.19.2.6. PININT_disable_falling_edge()

```
static void PININT_disable_falling_edge (
```

```
            uint8_t channel )  [inline], [static]
```

Inhabilitar detecciones por flanco descendente.

Parámetros

in	*channel*	Canal deseado

7.19.2.7. PININT_enable_high_level()

```
static void PININT_enable_high_level (
            uint8_t channel )  [inline], [static]
```

Habilitar detecciones por nivel alto.

Parámetros

in	*channel*	Canal deseado

7.19.2.8. PININT_disable_high_level()

```
static void PININT_disable_high_level (
            uint8_t channel )  [inline], [static]
```

Inhabilitar detecciones por flanco ascendente.

Parámetros

in	*channel*	Canal deseado

7.19.2.9. PININT_get_rising_edge_active()

```
static uint8_t PININT_get_rising_edge_active (
            void  )  [inline], [static]
```

Obtener interrupciones activas por flanco ascendente.

Devuelve

Mascara de bits con los canales activos

7.19.2.10. PININT_get_falling_edge_active()

```
static uint8_t PININT_get_falling_edge_active (
            void  )  [inline], [static]
```

Obtener interrupciones activas por flanco descendente.

Devuelve

Mascara de bits con los canales activos

7.19.2.11. PININT_get_level_active()

```
static uint8_t PININT_get_level_active (
            void  )  [inline], [static]
```

Obtener interrupciones activas por nivel.

Devuelve

Mascara de bits con los canales activos

7.19.2.12. PININT_clear_edge_level_irq()

```
static void PININT_clear_edge_level_irq (
            uint8_t channel )  [inline], [static]
```

Limpiar flag de interrupcion por flanco.

Parámetros

in	*channel*	Canal de interrupcion a limpiar

7.19.2.13. PININT_toggle_active_level()

```
static void PININT_toggle_active_level (
            uint8_t channel )  [inline], [static]
```

Invertir nivel activo de itnerrupcion por nivel.

Parámetros

in	*channel*	Canal a invertir

7.19.2.14. PININT_get_pattern_match_state()

```
static uint8_t PININT_get_pattern_match_state (
            void  )  [inline], [static]
```

Obtener estado actual del pattern match.

Devuelve

Mascara de bits indicando que miniterminos estan activos

7.19.2.15. PININT_config_pattern_match_source()

```
static void PININT_config_pattern_match_source (
            uint8_t channel,
            uint8_t slice )  [inline], [static]
```

Configuracion de las fuentes del pattern match.

Parámetros

in	*channel*	Entrada a configurar
in	*slice*	Slice a configurar

7.19.2.16. PINITN_enable_slice_as_endpoint()

```
static void PINITN_enable_slice_as_endpoint (
            uint8_t slice )  [inline], [static]
```

Habilitar slice como endpoint.

Parámetros

in	*slice*	Slice a configurar

7.19.2.17. PINITN_disable_slice_as_endpoint()

```
static void PINITN_disable_slice_as_endpoint (
            uint8_t slice )  [inline], [static]
```

Inhabilitar slice como endpoint.

Parámetros

in	*slice*	Slice a configurar

7.19.2.18. PINITN_config_slice_mode()

```
static void PINITN_config_slice_mode (
             uint8_t slice,
             PININT_match_contribution_en mode )  [inline], [static]
```

Configurar modo del slice.

Parámetros

in	*slice*	Slice a configurar

7.20. Referencia del Archivo includes/hpl/HPL_PMU.h

Declaraciones a nivel de abstraccion de periferico del PMU (LPC845)

```
#include <HRI_PMU.h>
```

Dependencia gráfica adjunta para HPL_PMU.h:

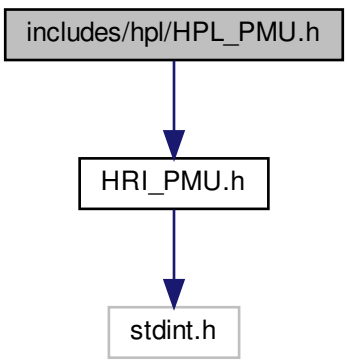
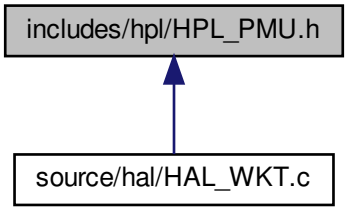


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **PMU_sleep_mode_en** { **PMU_SLEEP_MODE_SLEEP** = 0, **PMU_SLEEP_MODE_DEEP_SLEEP** }
- enum **PMU_power_mode_en** { **PMU_POWER_MODE_DEFAULT** = 0, **PMU_POWER_MODE_DEEP_S↩ LEEP**, **PMU_POWER_MODE_POWER_DOWN**, **PMU_POWER_MODE_DEEP_POWER_DOWN** }
- enum **PMU_general_purpouse_regiter_en** { **PMU_GENERAL_PURPOUSE_REGISTER_0** = 0, **PMU_↩ GENERAL_PURPOUSE_REGISTER_1**, **PMU_GENERAL_PURPOUSE_REGISTER_2**, **PMU_GENERA↩ L_PURPOUSE_REGISTER_3** }

Funciones

- static void PMU_set_sleep_on_exit (void)

  Activar funcion sleep on exit.
- static void PMU_clear_sleep_on_exit (void)

  Desactivar funcion sleep on exit.
- static void PMU_config_sleep_mode (PMU_sleep_mode_en sleep_mode)

  Configurar profundidad del sleep.
- static void **PMU_set_send_event_on_pending_bit** (void)
- static void **PMU_clear_send_event_on_pending_bit** (void)
- static void PMU_config_power_mode (PMU_power_mode_en power_mode)

  Configurar modo de energia en WFI.
- static void PMU_set_prevent_deep_power (void)

  Inhabilitar deep power-down mode en WFI.
- static uint8_t **PMU_get_sleep_flag** (void)
- static void **PMU_clear_sleep_flag** (void)
- static uint8_t **PMU_get_deep_power_down_flag** (void)
- static void **PMU_clear_deep_power_down_flag** (void)
- static void PMU_write_general_purpouse_register (PMU_general_purpouse_regiter_en reg, uint32_t data)

  Escibir un dato en un registro de proposito general.
- static uint32_t PMU_read_general_purpouse_register (PMU_general_purpouse_regiter_en reg)

  Leer un dato en un registro de proposito general.
- static void PMU_enable_wake_up_pin_hysteresis (void)

  Habilitar histeresis en el pin wake up.
- static void PMU_disable_wake_up_pin_hysteresis (void)

  Inhabilitar histeresis en el pin wake up.

- static void PMU_enable_wake_up_pin (void)
  Habilitar wake up pin.
- static void PMU_disable_wake_up_pin (void)
  Inhabilitar wake up pin.
- static void PMU_enable_low_power_oscillator (void)
  Habilitar low-power oscillator.
- static void PMU_disable_low_power_oscillator (void)
  Inhabilitar low-power oscillator.
- static void PMU_enable_low_power_oscillator_in_dpdmode (void)
  Habilitar low-power oscillator en deep power-down mode.
- static void PMU_disable_low_power_oscillator_in_dpdmode (void)
  Inhabilitar low-power oscillator en deep power-down mode.
- static void **PMU_enable_wake_up_clock_hysteresis** (void)
- static void **PMU_disable_wake_up_clock_hysteresis** (void)
- static void PMU_enable_wake_up_clock_pin (void)
  Habilitar el pin del clock externo de wake up.
- static void PMU_disable_wake_up_clock_pin (void)
  Inhabilitar el pin del clock externo de wake up.
- static void PMU_enable_reset_hysteresis (void)
  Habilitar histeresis en el pin de reset.
- static void PMU_disable_reset_hysteresis (void)
  Inhabilitar histeresis en el pin de reset.
- static void PMU_enable_reset (void)
  Habilitar funcion de reset en el pin.
- static void PMU_disable_reset (void)
  Inhabilitar funcion de reset en el pin.

Variables

- volatile SCR_reg_t ∗const SCR
  Registro SCR.
- volatile PMU_per_t ∗const PMU
  Periferico PMU.

7.20.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del PMU (LPC845)

Autor

Augusto Santini

Fecha

4/2020

Versión

1.0

7.20.2. Documentación de las funciones

7.20.2.1. PMU_config_sleep_mode()

```
static void PMU_config_sleep_mode (
            PMU_sleep_mode_en sleep_mode )  [inline], [static]
```

Configurar profundidad del sleep.

Parámetros

in	*sleep_mode*	Profundidad del sleep deseada

7.20.2.2. PMU_config_power_mode()

```
static void PMU_config_power_mode (
            PMU_power_mode_en power_mode )  [inline], [static]
```

Configurar modo de energia en WFI.

Parámetros

in	*power_mode*	Modo de energia deseado

7.20.2.3. PMU_set_prevent_deep_power()

```
static void PMU_set_prevent_deep_power (
            void  )  [inline], [static]
```

Inhabilitar deep power-down mode en WFI.

Inhabilita el deep power-down mode, aunque este configurado en ese modo, hasta el proximo reset

7.20.2.4. PMU_write_general_purpouse_register()

```
static void PMU_write_general_purpouse_register (
            PMU_general_purpouse_regiter_en reg,
            uint32_t data )  [inline], [static]
```

Escibir un dato en un registro de proposito general.

Parámetros

in	*reg*	Numero de registro a escribir
in	*data*	Dato a escribir en el registro de proposito general

7.20.2.5. PMU_read_general_purpouse_register()

```
static uint32_t PMU_read_general_purpouse_register (
            PMU_general_purpouse_regiter_en reg )  [inline], [static]
```

Leer un dato en un registro de proposito general.

Parámetros

in	*reg*	Numero de registro a escribir

Devuelve

Dato en el registro de proposito general

7.21. Referencia del Archivo includes/hpl/HPL_SPI.h

Declaraciones a nivel de abstraccion de periferico del SPI (LPC845)

```
#include <HRI_SPI.h>
```

Dependencia gráfica adjunta para HPL_SPI.h:

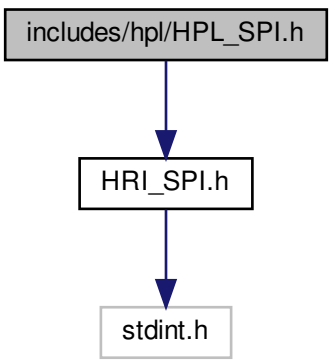
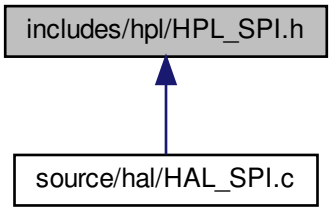


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **SPI_status_flag_en** {
  SPI_STATUS_FLAG_RXRDY = 0, **SPI_STATUS_FLAG_TXRDY**, **SPI_STATUS_FLAG_RXOV**, **SPI_STATUS_FLAG_TXUR**,
  SPI_STATUS_FLAG_SSA, **SPI_STATUS_FLAG_SSD**, **SPI_STATUS_FLAG_STALLED**, **SPI_STATUS_FLAG_ENDTRANSFER**,
  SPI_STATUS_FLAG_MSTIDLE }
- enum **SPI_irq_sel_en** {
  SPI_IRQ_RXRDY = 0, **SPI_IRQ_TXRDY**, **SPI_IRQ_RXOV**, **SPI_IRQ_TXUR**,
  SPI_IRQ_SSA, **SPI_IRQ_SSD** }
- enum **SPI_data_length_en** {
  SPI_DATA_LENGTH_1_BIT = 0, **SPI_DATA_LENGTH_2_BIT**, **SPI_DATA_LENGTH_3_BIT**, **SPI_DATA_LENGTH_4_BIT**,
  SPI_DATA_LENGTH_5_BIT, **SPI_DATA_LENGTH_6_BIT**, **SPI_DATA_LENGTH_7_BIT**, **SPI_DATA_LENGTH_8_BIT**,
  SPI_DATA_LENGTH_9_BIT, **SPI_DATA_LENGTH_10_BIT**, **SPI_DATA_LENGTH_11_BIT**, **SPI_DATA_LENGTH_12_BIT**,
  SPI_DATA_LENGTH_13_BIT, **SPI_DATA_LENGTH_14_BIT**, **SPI_DATA_LENGTH_15_BIT**, **SPI_DATA_LENGTH_16_BIT** }

Funciones

- static void SPI_enable (uint8_t inst)

  Habilitar modulo.
- static void SPI_disable (uint8_t inst)

  Inhabilitar modulo.
- static void SPI_set_master_mode (uint8_t inst)

  Configurar como modo master.
- static void SPI_set_slave_mode (uint8_t inst)

  Configurar como modo slave.
- static void SPI_set_data_order_msb_first (uint8_t inst)

  Configurar orden de datos MSB primero.
- static void SPI_set_data_order_lsb_first (uint8_t inst)

  Configurar orden de datos LSB primero.
- static void SPI_set_cpha_change (uint8_t inst)

Configurar fase del clock (modo change)

- static void SPI_set_cpha_capture (uint8_t inst)

  Configurar fase del clock (modo capture)
- static void SPI_set_cpol_low (uint8_t inst)

  Configurar polaridad del clock (polaridad baja)
- static void SPI_set_cpol_high (uint8_t inst)

  Configurar polaridad del clock (polaridad alta)
- static void SPI_enable_loopback_mode (uint8_t inst)

  Habilitar modo loopback.
- static void SPI_disable_loopback_mode (uint8_t inst)

  Inhabilitar modo loopback.
- static void SPI_set_ssel_active_low (uint8_t inst, uint8_t channel)

  Fijar polaridad de slave select como activa baja.
- static void SPI_set_ssel_active_high (uint8_t inst, uint8_t channel)

  Fijar polaridad de slave select como activa alta.
- static void SPI_set_pre_delay (uint8_t inst, uint8_t delay)

  Configurar ciclos de clock entre la activacion de SSEL y la transmision de datos.
- static void SPI_set_post_delay (uint8_t inst, uint8_t delay)

  Configurar ciclos de clock entre la finalizacion de transmision y desactivacion de SSEL.
- static void SPI_set_frame_delay (uint8_t inst, uint8_t delay)

  Configurar ciclos de clock entre transmisiones sin desactivar SSEL.
- static void SPI_set_transfer_delay (uint8_t inst, uint8_t delay)

  Configurar ciclos de clock entre desactivacion/activacion de SSEL.
- static uint8_t SPI_get_status_flag (uint8_t inst, SPI_status_flag_en flag)

  Leer un flag de status.
- static uint8_t SPI_clear_status_flag (uint8_t inst, SPI_status_flag_en flag)

  Limpiar un flag de status.
- static uint8_t SPI_enable_irq (uint8_t inst, SPI_irq_sel_en irq)

  Habilitar interrupcion.
- static uint8_t SPI_disable_irq (uint8_t inst, SPI_irq_sel_en irq)

  Inhabilitar interrupcion.
- static uint16_t SPI_read_rx_data (uint8_t inst)

  Leer resultado de la recepcion.
- static uint8_t SPI_get_active_ssl (uint8_t inst)

  Obtener slave select activo.
- static uint8_t SPI_get_sot_flag (uint8_t inst)

  Obtener estado del flag de start of transfer.
- static void SPI_write_txdata (uint8_t inst, uint16_t data)

  Escribir registro de datos de transmision.
- static void SPI_select_slave (uint8_t inst, uint8_t channel)

  Habilitar slave select para la proxima transmision.
- static void SPI_set_end_of_transmission (uint8_t inst)

  Indicar fin de transmision en la proxima transmision.
- static void SPI_clear_end_of_transmission (uint8_t inst)

  Limpiar fin de transmision en la proxima transmision.
- static void SPI_set_end_of_frame (uint8_t inst)

  Indicar fin de trama en la proxima transmision.
- static void SPI_clear_end_of_frame (uint8_t inst)

  Limpiar fin de trama en la proxima transmision.
- static void SPI_set_rx_ignore (uint8_t inst)

  Ignorar recepcion en la proxima transmision.

- static void SPI_clear_rx_ignore (uint8_t inst)

  No ignorar recepcino en la proxima transmision.
- static void SPI_set_data_length (uint8_t inst, SPI_data_length_en data_length)

  Configurar largo de bits de palabra.
- static void SPI_set_data_and_control (uint8_t inst, SPI_TXDATCTL_reg_t *data_and_control)

  Escribir data a transmitir y control al mismo tiempo (en una unica escritura)
- static void SPI_set_clock_div (uint8_t inst, uint16_t div)

  Configurar divisor de clock.
- static uint8_t SPI_get_irq_flag_status (uint8_t inst, SPI_irq_sel_en irq)

  Leer flag de interrupcion actual.

Variables

- volatile SPI_per_t *const SPI [ ]

  Perifericos SPI.

7.21.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del SPI (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.21.2. Documentación de las funciones

7.21.2.1. SPI_enable()

```
static void SPI_enable (
            uint8_t inst )  [inline], [static]
```

Habilitar modulo.

Parámetros

in	*inst*	Instancia a habilitar

7.21.2.2. SPI_disable()

```
static void SPI_disable (
            uint8_t inst )  [inline], [static]
```

Inhabilitar modulo.

Parámetros

in	*inst*	Instancia a inhabilitar

7.21.2.3. SPI_set_master_mode()

```
static void SPI_set_master_mode (
            uint8_t inst )  [inline], [static]
```

Configurar como modo master.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.4. SPI_set_slave_mode()

```
static void SPI_set_slave_mode (
            uint8_t inst )  [inline], [static]
```

Configurar como modo slave.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.5. SPI_set_data_order_msb_first()

```
static void SPI_set_data_order_msb_first (
            uint8_t inst )  [inline], [static]
```

Configurar orden de datos MSB primero.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.6. SPI_set_data_order_lsb_first()

```
static void SPI_set_data_order_lsb_first (
            uint8_t inst )  [inline], [static]
```

Configurar orden de datos LSB primero.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.7. SPI_set_cpha_change()

```
static void SPI_set_cpha_change (
            uint8_t inst )  [inline], [static]
```

Configurar fase del clock (modo change)

Parámetros

in	*inst*	Instancia a configurar

7.21.2.8. SPI_set_cpha_capture()

```
static void SPI_set_cpha_capture (
            uint8_t inst )  [inline], [static]
```

Configurar fase del clock (modo capture)

Parámetros

in	*inst*	Instancia a configurar

7.21.2.9. SPI_set_cpol_low()

```
static void SPI_set_cpol_low (
            uint8_t inst )  [inline], [static]
```

Configurar polaridad del clock (polaridad baja)

Parámetros

in	*inst*	Instancia a configurar

7.21.2.10. SPI_set_cpol_high()

```
static void SPI_set_cpol_high (
            uint8_t inst )  [inline], [static]
```

Configurar polaridad del clock (polaridad alta)

Parámetros

in	*inst*	Instancia a configurar

7.21.2.11. SPI_enable_loopback_mode()

```
static void SPI_enable_loopback_mode (
            uint8_t inst )  [inline], [static]
```

Habilitar modo loopback.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.12. SPI_disable_loopback_mode()

```
static void SPI_disable_loopback_mode (
            uint8_t inst )  [inline], [static]
```

Inhabilitar modo loopback.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.13. SPI_set_ssel_active_low()

```
static void SPI_set_ssel_active_low (
            uint8_t inst,
            uint8_t channel )  [inline], [static]
```

Fijar polaridad de slave select como activa baja.

Parámetros

in	*inst*	Instancia a configurar
in	*channel*	Canal de SSEL a configurar

7.21.2.14. SPI_set_ssel_active_high()

```
static void SPI_set_ssel_active_high (
            uint8_t inst,
            uint8_t channel )  [inline], [static]
```

Fijar polaridad de slave select como activa alta.

Parámetros

in	*inst*	Instancia a configurar
in	*channel*	Canal de SSEL a configurar

7.21.2.15. SPI_set_pre_delay()

```
static void SPI_set_pre_delay (
            uint8_t inst,
            uint8_t delay )  [inline], [static]
```

Configurar ciclos de clock entre la activacion de SSEL y la transmision de datos.

Parámetros

in	*inst*	Instancia a configurar
in	*delay*	Clocks de delay deseados

7.21.2.16. SPI_set_post_delay()

```
static void SPI_set_post_delay (
            uint8_t inst,
            uint8_t delay )  [inline], [static]
```

Configurar ciclos de clock entre la finalizacion de transmision y desactivacion de SSEL.

Parámetros

in	*inst*	Instancia a configurar
in	*delay*	Clocks de delay deseados

7.21.2.17. SPI_set_frame_delay()

```
static void SPI_set_frame_delay (
            uint8_t inst,
            uint8_t delay )  [inline], [static]
```

Configurar ciclos de clock entre transmisiones sin desactivar SSEL.

Parámetros

in	*inst*	Instancia a configurar
in	*delay*	Clocks de delay deseados

7.21.2.18. SPI_set_transfer_delay()

```
static void SPI_set_transfer_delay (
            uint8_t inst,
            uint8_t delay )  [inline], [static]
```

Configurar ciclos de clock entre desactivacion/activacion de SSEL.

Parámetros

in	*inst*	Instancia a configurar
in	*delay*	Clocks de delay deseados

7.21.2.19. SPI_get_status_flag()

```
static uint8_t SPI_get_status_flag (
            uint8_t inst,
            SPI_status_flag_en flag )  [inline], [static]
```

Leer un flag de status.

Parámetros

in	*inst*	Instancia a consultar
in	*flag*	Flag a consultar

Devuelve

Estado del flag actual

7.21.2.20. SPI_clear_status_flag()

```
static uint8_t SPI_clear_status_flag (
            uint8_t inst,
            SPI_status_flag_en flag )  [inline], [static]
```

Limpiar un flag de status.

Parámetros

in	*inst*	Instancia a limpiar
in	*flag*	Flag a limpiar

7.21.2.21. SPI_enable_irq()

```
static uint8_t SPI_enable_irq (
            uint8_t inst,
            SPI_irq_sel_en irq )  [inline], [static]
```

Habilitar interrupcion.

Parámetros

in	*inst*	Instancia a configurar
in	*irq*	Interrupcion a habilitar

7.21.2.22. SPI_disable_irq()

```
static uint8_t SPI_disable_irq (
            uint8_t inst,
            SPI_irq_sel_en irq )  [inline], [static]
```

Inhabilitar interrupcion.

Parámetros

in	*inst*	Instancia a configurar
in	*irq*	Interrupcion a inhabilitar

7.21.2.23. SPI_read_rx_data()

```
static uint16_t SPI_read_rx_data (
            uint8_t inst )  [inline], [static]
```

Leer resultado de la recepcion.

Parámetros

in	*inst*	Instancia a consultar

7.21.2.24. SPI_get_active_ssl()

```
static uint8_t SPI_get_active_ssl (
            uint8_t inst )  [inline], [static]
```

Obtener slave select activo.

Parámetros

in	*inst*	Instancia a consultar

7.21.2.25. SPI_get_sot_flag()

```
static uint8_t SPI_get_sot_flag (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag de start of transfer.

Parámetros

in	*inst*	Instancia a consultar

7.21.2.26. SPI_write_txdata()

```
static void SPI_write_txdata (
            uint8_t inst,
            uint16_t data )  [inline], [static]
```

Escribir registro de datos de transmision.

Parámetros

in	*inst*	Instancia a escribir
in	*data*	Dato a escribir

7.21.2.27. SPI_select_slave()

```
static void SPI_select_slave (
            uint8_t inst,
            uint8_t channel )  [inline], [static]
```

Habilitar slave select para la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar
in	*channel*	Canal a utilizar en la proxima transmision

7.21.2.28. SPI_set_end_of_transmission()

```
static void SPI_set_end_of_transmission (
            uint8_t inst )  [inline], [static]
```

Indicar fin de transmision en la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.29. SPI_clear_end_of_transmission()

```
static void SPI_clear_end_of_transmission (
            uint8_t inst )  [inline], [static]
```

Limpiar fin de transmision en la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.30. SPI_set_end_of_frame()

```
static void SPI_set_end_of_frame (
            uint8_t inst )  [inline], [static]
```

Indicar fin de trama en la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.31. SPI_clear_end_of_frame()

```
static void SPI_clear_end_of_frame (
            uint8_t inst )  [inline], [static]
```

Limpiar fin de trama en la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.32. SPI_set_rx_ignore()

```
static void SPI_set_rx_ignore (
            uint8_t inst )  [inline], [static]
```

Ignorar recepcion en la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.33. SPI_clear_rx_ignore()

```
static void SPI_clear_rx_ignore (
            uint8_t inst )  [inline], [static]
```

No ignorar recepcino en la proxima transmision.

Parámetros

in	*inst*	Instancia a configurar

7.21.2.34. SPI_set_data_length()

```
static void SPI_set_data_length (
            uint8_t inst,
            SPI_data_length_en data_length )  [inline], [static]
```

Configurar largo de bits de palabra.

Parámetros

in	*inst*	Instancia a configurar
in	*data_length*	Largo de palabra deseado

7.21.2.35. SPI_set_data_and_control()

```
static void SPI_set_data_and_control (
            uint8_t inst,
            SPI_TXDATCTL_reg_t * data_and_control )  [inline], [static]
```

Escribir data a transmitir y control al mismo tiempo (en una unica escritura)

Parámetros

in	*inst*	Instancia a utilizar
in	*data_and_control*	Dato a transmitir y control

7.21.2.36. SPI_set_clock_div()

```
static void SPI_set_clock_div (
            uint8_t inst,
            uint16_t div )  [inline], [static]
```

Configurar divisor de clock.

Parámetros

in	*inst*	Instancia a configurar
in	*div*	Divisor deseado (el valor efectivo es este valor +1)

7.21.2.37. SPI_get_irq_flag_status()

```
static uint8_t SPI_get_irq_flag_status (
            uint8_t inst,
            SPI_irq_sel_en irq )  [inline], [static]
```

Leer flag de interrupcion actual.

Parámetros

in	*inst*	Instancia a consultar
in	*irq*	Flag de interrupcion a consultar

7.22. Referencia del Archivo includes/hpl/HPL_SWM.h

Definiciones a nivel de periferico del modulo SWM (LPC845)

```
#include <HRI_SWM.h>
#include <HPL_SYSCON.h>
```

Dependencia gráfica adjunta para HPL_SWM.h:

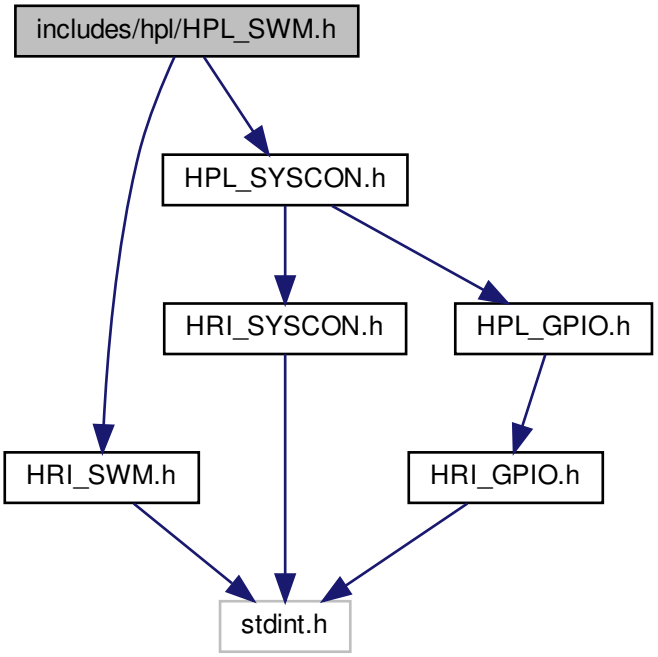
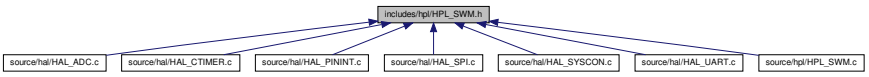


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **SWM_enable_en** { **SWM_ENABLE** = 0, **SWM_DISABLE** }

Funciones

- static void SWM_init (void)

  Inicializacion de la Switch Matrix.
- static void SWM_deinit (void)

  Deinicializacion de la Switch Matrix.
- static void SWM_assign_uart_TXD (uint8_t uart, uint8_t port, uint8_t pin)

  Asignar un pin del MCU a la funcion UARTn TXD.
- static void SWM_assign_uart_RXD (uint8_t uart, uint8_t port, uint8_t pin)

  Asignar un pin del MCU a la funcion UARTn RXD.

- static void SWM_assign_uart_RTS (uint8_t uart, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion UARTn RTS.
- static void SWM_assign_uart_CTS (uint8_t uart, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion UARTn CTS.
- static void SWM_assign_uart_SCLK (uint8_t uart, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion UARTn SCLK.
- static void SWM_assign_spi_SCK (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn SCK.
- static void SWM_assign_spi_MOSI (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn MOSI.
- static void SWM_assign_spi_MISO (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn MISO.
- static void SWM_assign_spi_SSEL0 (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn SSEL0.
- static void SWM_assign_spi_SSEL1 (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn SSEL1.
- static void SWM_assign_spi_SSEL2 (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn SSEL2.
- static void SWM_assign_spi_SSEL3 (uint8_t spi, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SPIn SSEL3.
- static void SWM_assign_sct_IN_A (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT IN_A.
- static void SWM_assign_sct_IN_B (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT IN_B.
- static void SWM_assign_sct_IN_C (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT IN_C.
- static void SWM_assign_sct_IN_D (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT IN_D.
- static void SWM_assign_sct_OUT0 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT0.
- static void SWM_assign_sct_OUT1 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT1.
- static void SWM_assign_sct_OUT2 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT2.
- static void SWM_assign_sct_OUT3 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT3.
- static void SWM_assign_sct_OUT4 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT4.
- static void SWM_assign_sct_OUT5 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT5.
- static void SWM_assign_sct_OUT6 (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion SCT OUT6.
- static void SWM_assign_iic_SDA (uint8_t iic, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion IICn SDA.
- static void SWM_assign_iic_SCL (uint8_t iic, uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion IICn SCL.
- static void SWM_assign_COMP0_OUT (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion COMP0 OUT.
- static void SWM_assign_CLKOUT (uint8_t port, uint8_t pin)
  Asignar un pin del MCU a la funcion CLKOUT.
- static void SWM_assign_INT_BMAT (uint8_t port, uint8_t pin)

Asignar un pin del MCU a la funcion INT BMAT.

- static void SWM_assign_T0_MAT (uint8_t mat, uint8_t port, uint8_t pin)

  Asignar un pin del MCU a la funcion T0 MATn.
- static void SWM_assign_T0_CAP (uint8_t cap, uint8_t port, uint8_t pin)

  Asignar un pin del MCU a la funcion T0 CAPn.
- static void SWM_enable_ACMP (uint8_t acmp, SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion ACMPn.
- static void SWM_enable_SWCLK (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion SWCLK.
- static void SWM_enable_SWDIO (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion SWDIO.
- static void SWM_enable_XTALIN (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion XTALIN.
- static void SWM_enable_XTALOUT (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion XTALOUT.
- static void SWM_enable_RESETN (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion RESETN.
- static void SWM_enable_CLKIN (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion CLKIN.
- static void SWM_enable_VDDCMP (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion VDDCMP.
- static void SWM_enable_ADC (uint8_t adc, SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion ADC.
- static void SWM_enable_DAC (uint8_t dac, SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion DAC.
- static void SWM_enable_CAPTX (uint8_t captx, SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion CAPTX.
- static void SWM_enable_CAPYL (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion CAPYL.
- static void SWM_enable_CAPYH (SWM_enable_en en_dis)

  Habilitar/inhabilitar la funcion CAPYH.

Variables

- volatile SWM_per_t ∗const SWM

  Periferico SWM.

7.22.1. Descripción detallada

Definiciones a nivel de periferico del modulo SWM (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.22.2. Documentación de las funciones

7.22.2.1. SWM_assign_uart_TXD()

```
static void SWM_assign_uart_TXD (
            uint8_t uart,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion UARTn TXD.

Parámetros

in	*uart*	Instancia de UART a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.2. SWM_assign_uart_RXD()

```
static void SWM_assign_uart_RXD (
            uint8_t uart,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion UARTn RXD.

Parámetros

in	*uart*	Instancia de UART a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.3. SWM_assign_uart_RTS()

```
static void SWM_assign_uart_RTS (
            uint8_t uart,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion UARTn RTS.

Parámetros

in	*uart*	Instancia de UART a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.4. SWM_assign_uart_CTS()

```
static void SWM_assign_uart_CTS (
            uint8_t uart,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion UARTn CTS.

Parámetros

in	*uart*	Instancia de UART a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.5. SWM_assign_uart_SCLK()

```
static void SWM_assign_uart_SCLK (
            uint8_t uart,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion UARTn SCLK.

Parámetros

in	*uart*	Instancia de UART a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.6. SWM_assign_spi_SCK()

```
static void SWM_assign_spi_SCK (
            uint8_t spi,
```

```
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn SCK.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.7. SWM_assign_spi_MOSI()

```
static void SWM_assign_spi_MOSI (
            uint8_t spi,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn MOSI.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.8. SWM_assign_spi_MISO()

```
static void SWM_assign_spi_MISO (
            uint8_t spi,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn MISO.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.9. SWM_assign_spi_SSEL0()

```
static void SWM_assign_spi_SSEL0 (
            uint8_t spi,
```

```
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn SSEL0.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.10. SWM_assign_spi_SSEL1()

```
static void SWM_assign_spi_SSEL1 (
            uint8_t spi,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn SSEL1.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.11. SWM_assign_spi_SSEL2()

```
static void SWM_assign_spi_SSEL2 (
            uint8_t spi,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn SSEL2.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.12. SWM_assign_spi_SSEL3()

```
static void SWM_assign_spi_SSEL3 (
            uint8_t spi,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SPIn SSEL3.

Parámetros

in	*spi*	Instancia de SPI a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.13. SWM_assign_sct_IN_A()

```
static void SWM_assign_sct_IN_A (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT IN_A.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.14. SWM_assign_sct_IN_B()

```
static void SWM_assign_sct_IN_B (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT IN_B.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.15. SWM_assign_sct_IN_C()

```
static void SWM_assign_sct_IN_C (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT IN_C.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.16. SWM_assign_sct_IN_D()

```
static void SWM_assign_sct_IN_D (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT IN_D.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.17. SWM_assign_sct_OUT0()

```
static void SWM_assign_sct_OUT0 (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT0.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.18. SWM_assign_sct_OUT1()

```
static void SWM_assign_sct_OUT1 (
```

```
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT1.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.19. SWM_assign_sct_OUT2()

```
static void SWM_assign_sct_OUT2 (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT2.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.20. SWM_assign_sct_OUT3()

```
static void SWM_assign_sct_OUT3 (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT3.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.21. SWM_assign_sct_OUT4()

```
static void SWM_assign_sct_OUT4 (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT4.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.22. SWM_assign_sct_OUT5()

```
static void SWM_assign_sct_OUT5 (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT5.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.23. SWM_assign_sct_OUT6()

```
static void SWM_assign_sct_OUT6 (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion SCT OUT6.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.24. SWM_assign_iic_SDA()

```
static void SWM_assign_iic_SDA (
            uint8_t iic,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion IICn SDA.

En el caso del IIC0, no se le da importancia al puerto y pin

Parámetros

in	*iic*	Instancia de IIC a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.25. SWM_assign_iic_SCL()

```
static void SWM_assign_iic_SCL (
            uint8_t iic,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion IICn SCL.

En el caso del IIC0, no se le da importancia al puerto y pin

Parámetros

in	*iic*	Instancia de IIC a la cual asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.26. SWM_assign_COMP0_OUT()

```
static void SWM_assign_COMP0_OUT (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion COMP0 OUT.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.27. SWM_assign_CLKOUT()

```
static void SWM_assign_CLKOUT (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion CLKOUT.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.28. SWM_assign_INT_BMAT()

```
static void SWM_assign_INT_BMAT (
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion INT BMAT.

Parámetros

in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.29. SWM_assign_T0_MAT()

```
static void SWM_assign_T0_MAT (
            uint8_t mat,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion T0 MATn.

Parámetros

in	*mat*	Numero de MATn a asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.30. SWM_assign_T0_CAP()

```
static void SWM_assign_T0_CAP (
            uint8_t cap,
            uint8_t port,
            uint8_t pin )  [inline], [static]
```

Asignar un pin del MCU a la funcion T0 CAPn.

Parámetros

in	*cap*	Numero de CAPn a asignar
in	*port*	Numero de puerto a asignar
in	*pin*	Numero de pin a asignar

7.22.2.31. SWM_enable_ACMP()

```
static void SWM_enable_ACMP (
            uint8_t acmp,
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion ACMPn.

Parámetros

in	*acmp*	Numero de ACMP a asignar
in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.32. SWM_enable_SWCLK()

```
static void SWM_enable_SWCLK (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion SWCLK.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.33. SWM_enable_SWDIO()

```
static void SWM_enable_SWDIO (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion SWDIO.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.34. SWM_enable_XTALIN()

```
static void SWM_enable_XTALIN (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion XTALIN.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.35. SWM_enable_XTALOUT()

```
static void SWM_enable_XTALOUT (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion XTALOUT.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.36. SWM_enable_RESETN()

```
static void SWM_enable_RESETN (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion RESETN.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.37. SWM_enable_CLKIN()

```
static void SWM_enable_CLKIN (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion CLKIN.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.38. SWM_enable_VDDCMP()

```
static void SWM_enable_VDDCMP (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion VDDCMP.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.39. SWM_enable_ADC()

```
static void SWM_enable_ADC (
            uint8_t adc,
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion ADC.

Parámetros

in	*adc*	Numero de ADC a asignar
in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.40. SWM_enable_DAC()

```
static void SWM_enable_DAC (
            uint8_t dac,
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion DAC.

Parámetros

in	*dac*	Numero de DAC a asignar
in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.41. SWM_enable_CAPTX()

```
static void SWM_enable_CAPTX (
            uint8_t captx,
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion CAPTX.

Parámetros

in	*captx*	Numero de CAPTX a asignar
in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.42. SWM_enable_CAPYL()

```
static void SWM_enable_CAPYL (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion CAPYL.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.22.2.43. SWM_enable_CAPYH()

```
static void SWM_enable_CAPYH (
            SWM_enable_en en_dis )  [inline], [static]
```

Habilitar/inhabilitar la funcion CAPYH.

Parámetros

in	*en_dis*	Seleccion de habilitacion o inhabilitacion

7.23. Referencia del Archivo includes/hpl/HPL_SYSCON.h

Declaraciones a nivel de abstraccion de periferico del SYSCON (LPC845)

```
#include <HRI_SYSCON.h>
#include <HPL_GPIO.h>
```

Dependencia gráfica adjunta para HPL_SYSCON.h:

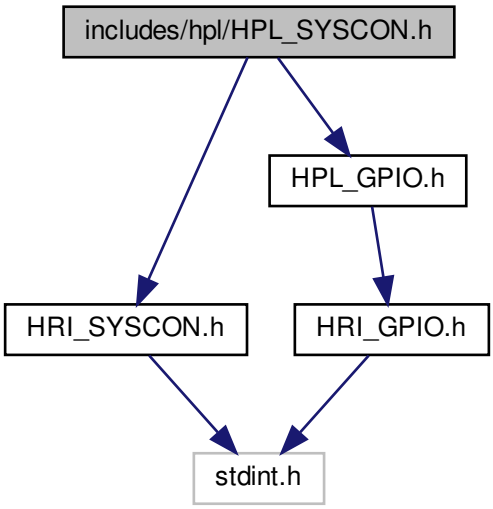
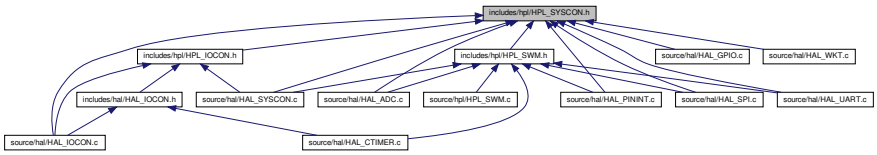


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **SYSCON_bypass_sel_en** { **SYSCON_BYPASS_DISABLED** = 0, **SYSCON_BYPASS_ENABLED** }
- enum **SYSCON_freqrange_sel_en** { **SYSCON_FREQRANGE_MINUS_20MHZ** = 0, **SYSCON_FREQRANGE_PLUS_20MHZ** }
- enum **SYSCON_watchdog_clkana_sel_en** { **SYSCON_WATCHDOG_CLKANA_0KHZ** = 0, **SYSCON_WATCHDOG_CLKANA_600KHZ**, **SYSCON_WATCHDOG_CLKANA_1050KHZ**, **SYSCON_WATCHDOG_CLKANA_1400KHZ**, **SYSCON_WATCHDOG_CLKANA_1750KHZ**, **SYSCON_WATCHDOG_CLKANA_2100KHZ**, **SYSCON_WATCHDOG_CLKANA_2400KHZ**, **SYSCON_WATCHDOG_CLKANA_3000KHZ**, **SYSCON_WATCHDOG_CLKANA_3250KHZ**, **SYSCON_WATCHDOG_CLKANA_3500KHZ**, **SYSCON_WATCHDOG_CLKANA_3750KHZ**, **SYSCON_WATCHDOG_CLKANA_4000KHZ**, **SYSCON_WATCHDOG_CLKANA_4200KHZ**, **SYSCON_WATCHDOG_CLKANA_4400KHZ**, **SYSCON_WATCHDOG_CLKANA_4600KHZ** }
- enum **SYSCON_pll_source_sel_en** { **SYSCON_PLL_SOURCE_SEL_FRO** = 0, **SYSCON_PLL_SOURCE_SEL_EXT_CLK**, **SYSCON_PLL_SOURCE_SEL_WATCHDOG**, **SYSCON_PLL_SOURCE_SEL_FRO_DIV** }

- enum **SYSCON_main_clock_sel_en** {
  SYSCON_MAIN_CLOCK_SEL_FRO = 0, **SYSCON_MAIN_CLOCK_SEL_EXT_CLK**, **SYSCON_MAIN_↩
  CLOCK_SEL_WATCHDOG**, **SYSCON_MAIN_CLOCK_SEL_FRO_DIV**,
  SYSCON_MAIN_CLOCK_SEL_PLL }
- enum **SYSCON_capacitive_clock_sel_en** {
  SYSCON_CAPACITIVE_CLOCK_SEL_FRO = 0, **SYSCON_CAPACITIVE_CLOCK_SEL_MAIN_CLOCK**,
  SYSCON_CAPACITIVE_CLOCK_SEL_SYS_PLL, **SYSCON_CAPACITIVE_CLOCK_SEL_FRO_DIV**,
  SYSCON_CAPACITIVE_CLOCK_SEL_WATCHDOG_OSC }
- enum **SYSCON_adc_clock_sel_en** { **SYSCON_ADC_CLOCK_SEL_FRO** = 0, **SYSCON_ADC_CLOCK↩
  _SEL_SYS_PLL** }
- enum **SYSCON_sct_clock_sel_en** { **SYSCON_SCT_CLOCK_SEL_FRO** = 0, **SYSCON_SCT_CLOCK_↩
  SEL_MAIN_CLOCK**, **SYSCON_SCT_CLOCK_SEL_SYS_PLL** }
- enum **SYSCON_ext_clock_source_sel_en** { **SYSCON_EXT_CLOCK_SOURCE_SEL_CRYSTAL** = 0, **S↩
  YSCON_EXT_CLOCK_SOURCE_SEL_CLK_IN** }
- enum **SYSCON_enable_clock_sel_en** {
  SYSCON_ENABLE_CLOCK_SEL_ROM = 1, **SYSCON_ENABLE_CLOCK_SEL_RAM**, **SYSCON_ENA↩
  BLE_CLOCK_SEL_FLASH** = 4, **SYSCON_ENABLE_CLOCK_SEL_IIC0**,
  SYSCON_ENABLE_CLOCK_SEL_GPIO0, **SYSCON_ENABLE_CLOCK_SEL_SWM**, **SYSCON_ENAB↩
  LE_CLOCK_SEL_SCT**, **SYSCON_ENABLE_CLOCK_SEL_WKT**,
  SYSCON_ENABLE_CLOCK_SEL_MRT, **SYSCON_ENABLE_CLOCK_SEL_SPI0**, **SYSCON_ENABLE↩
  _CLOCK_SEL_SPI1**, **SYSCON_ENABLE_CLOCK_SEL_CRC**,
  SYSCON_ENABLE_CLOCK_SEL_UART0, **SYSCON_ENABLE_CLOCK_SEL_UART1**, **SYSCON_ENA↩
  BLE_CLOCK_SEL_UART2**, **SYSCON_ENABLE_CLOCK_SEL_WWDT**,
  SYSCON_ENABLE_CLOCK_SEL_IOCON, **SYSCON_ENABLE_CLOCK_SEL_ACMP**, **SYSCON_ENA↩
  BLE_CLOCK_SEL_GPIO1**, **SYSCON_ENABLE_CLOCK_SEL_IIC1**,
  SYSCON_ENABLE_CLOCK_SEL_IIC2, **SYSCON_ENABLE_CLOCK_SEL_IIC3**, **SYSCON_ENABLE_↩
  CLOCK_SEL_ADC**, **SYSCON_ENABLE_CLOCK_SEL_CTIMER**,
  SYSCON_ENABLE_CLOCK_SEL_MTB, **SYSCON_ENABLE_CLOCK_SEL_DAC0**, **SYSCON_ENABL↩
  E_CLOCK_SEL_GPIO_INT**, **SYSCON_ENABLE_CLOCK_SEL_DMA**,
  SYSCON_ENABLE_CLOCK_SEL_UART3, **SYSCON_ENABLE_CLOCK_SEL_UART4**, **SYSCON_ENA↩
  BLE_CLOCK_SEL_CAPT**, **SYSCON_ENABLE_CLOCK_SEL_DAC1** }
- enum **SYSCON_reset_sel_en** {
  SYSCON_RESET_SEL_FLASH = 4, **SYSCON_RESET_SEL_IIC0**, **SYSCON_RESET_SEL_GPIO0**, **SY↩
  SCON_RESET_SEL_SWM**,
  SYSCON_RESET_SEL_SCT, **SYSCON_RESET_SEL_WKT**, **SYSCON_RESET_SEL_MRT**, **SYSCON_↩
  RESET_SEL_SPI0**,
  SYSCON_RESET_SEL_SPI1, **SYSCON_RESET_SEL_CRC**, **SYSCON_RESET_SEL_UART0**, **SYSCO↩
  N_RESET_SEL_UART1**,
  SYSCON_RESET_SEL_UART2, **SYSCON_RESET_SEL_IOCON** = 18, **SYSCON_RESET_SEL_ACMP**,
  SYSCON_RESET_SEL_GPIO1,
  SYSCON_RESET_SEL_IIC1, **SYSCON_RESET_SEL_IIC2**, **SYSCON_RESET_SEL_IIC3**, **SYSCON_RE↩
  SET_SEL_ADC**,
  SYSCON_RESET_SEL_CTIMER, **SYSCON_RESET_SEL_DAC0** = 27, **SYSCON_RESET_SEL_GPIOINT**,
  SYSCON_RESET_SEL_DMA,
  SYSCON_RESET_SEL_UART3, **SYSCON_RESET_SEL_UART4**, **SYSCON_RESET_SEL_CAPT**, **SYS↩
  CON_RESET_SEL_DAC1**,
  SYSCON_RESET_SEL_FRG0 = 35, **SYSCON_RESET_SEL_FRG1** }
- enum **SYSCON_peripheral_sel_en** {
  SYSCON_PERIPHERAL_SEL_UART0 = 0, **SYSCON_PERIPHERAL_SEL_UART1**, **SYSCON_PERIPH↩
  ERAL_SEL_UART2**, **SYSCON_PERIPHERAL_SEL_UART3**,
  SYSCON_PERIPHERAL_SEL_UART4, **SYSCON_PERIPHERAL_SEL_I2C0**, **SYSCON_PERIPHERAL_↩
  SEL_I2C1**, **SYSCON_PERIPHERAL_SEL_I2C2**,
  SYSCON_PERIPHERAL_SEL_I2C3, **SYSCON_PERIPHERAL_SEL_SPI0**, **SYSCON_PERIPHERAL_S↩
  EL_SPI1** }
- enum **SYSCON_peripheral_clock_sel_en** {
  SYSCON_PERIPHERAL_CLOCK_SEL_FRO = 0, **SYSCON_PERIPHERAL_CLOCK_SEL_MAIN**, **SYSC↩
  ON_PERIPHERAL_CLOCK_SEL_FRG0**, **SYSCON_PERIPHERAL_CLOCK_SEL_FRG1**,
  SYSCON_PERIPHERAL_CLOCK_SEL_FRO_DIV, **SYSCON_PERIPHERAL_CLOCK_SEL_NONE** = 7 }

- enum **SYSCON_frg_clock_sel_en** { **SYSCON_FRG_CLOCK_SEL_FRO** = 0, **SYSCON_FRG_CLOCK_SEL_MAIN_CLOCK**, **SYSCON_FRG_CLOCK_SEL_SYS_PLL**, **SYSCON_FRG_CLOCK_SEL_NONE** }
- enum **SYSCON_clkout_source_sel_en** { **SYSCON_CLKOUT_SOURCE_SEL_FRO** = 0, **SYSCON_CLKOUT_SOURCE_SEL_MAIN_CLOCK**, **SYSCON_CLKOUT_SOURCE_SEL_SYS_PLL**, **SYSCON_CLKOUT_SOURCE_SEL_EXT_CLOCK**, **SYSCON_CLKOUT_SOURCE_SEL_WATCHDOG_OSC** }
- enum **SYSCON_bod_level_en** { **SYSCON_BOD_LEVEL_1** = 1, **SYSCON_BOD_LEVEL_2**, **SYSCON_BOD_LEVEL_3** }
- enum **SYSCON_bod_enale_en** { **SYSCON_BOD_DISABLE** = 0, **SYSCON_BOD_ENABLE** }
- enum **SYSCON_nmi_enable_en** { **SYSCON_NMI_DISABLE** = 0, **SYSCON_NMI_ENABLE** }
- enum **SYSCON_enable_wakeup_sel_en** { **SYSCON_WAKEUP_ENABLE_SEL_PINT0** = 0, **SYSCON_WAKEUP_ENABLE_SEL_PINT1**, **SYSCON_WAKEUP_ENABLE_SEL_PINT2**, **SYSCON_WAKEUP_ENABLE_SEL_PINT3**, **SYSCON_WAKEUP_ENABLE_SEL_PINT4**, **SYSCON_WAKEUP_ENABLE_SEL_PINT5**, **SYSCON_WAKEUP_ENABLE_SEL_PINT6**, **SYSCON_WAKEUP_ENABLE_SEL_PINT7**, **SYSCON_WAKEUP_ENABLE_SEL_SPI0** = 32, **SYSCON_WAKEUP_ENABLE_SEL_SPI1**, **SYSCON_WAKEUP_ENABLE_SEL_USART0** = 35, **SYSCON_WAKEUP_ENABLE_SEL_USART1**, **SYSCON_WAKEUP_ENABLE_SEL_USART2**, **SYSCON_WAKEUP_ENABLE_SEL_IIC1** = 39, **SYSCON_WAKEUP_ENABLE_SEL_IIC0**, **SYSCON_WAKEUP_ENABLE_SEL_CAPTOUCH** = 43, **SYSCON_WAKEUP_ENABLE_SEL_WWDT**, **SYSCON_WAKEUP_ENABLE_SEL_BOD**, **SYSCON_WAKEUP_ENABLE_SEL_WKT** = 47, **SYSCON_WAKEUP_ENABLE_SEL_IIC2** = 53, **SYSCON_WAKEUP_ENABLE_SEL_IIC3**, **SYSCON_WAKEUP_ENABLE_SEL_USART3** = 62, **SYSCON_WAKEUP_ENABLE_SEL_USART4** }
- enum **SYSCON_deep_sleep_power_en** { **SYSCON_DEEP_SLEEP_POWERED** = 0, **SYSCON_DEEP_SLEEP_POWERED_DOWN** }
- enum **SYSCON_wakeup_power_sel_en** { **SYSCON_WAKEUP_POWER_SEL_FROOUT** = 0, **SYSCON_WAKEUP_POWER_SEL_FRO**, **SYSCON_WAKEUP_POWER_SEL_FLASH**, **SYSCON_WAKEUP_POWER_SEL_BOD**, **SYSCON_WAKEUP_POWER_SEL_ADC**, **SYSCON_WAKEUP_POWER_SEL_SYSOSC**, **SYSCON_WAKEUP_POWER_SEL_WDTOSC**, **SYSCON_WAKEUP_POWER_SEL_SYSPLL**, **SYSCON_WAKEUP_POWER_SEL_VREF2** = 10, **SYSCON_WAKEUP_POWER_SEL_DAC0** = 13, **SYSCON_WAKEUP_POWER_SEL_DAC1**, **SYSCON_WAKEUP_POWER_SEL_ACMP** }
- enum **SYSCON_power_sel_en** { **SYSCON_POWER_SEL_FROOUT** = 0, **SYSCON_POWER_SEL_FRO**, **SYSCON_POWER_SEL_FLASH**, **SYSCON_POWER_SEL_BOD**, **SYSCON_POWER_SEL_ADC**, **SYSCON_POWER_SEL_SYSOSC**, **SYSCON_POWER_SEL_WDTOSC**, **SYSCON_POWER_SEL_SYSPLL**, **SYSCON_POWER_SEL_DAC0** = 13, **SYSCON_POWER_SEL_DAC1**, **SYSCON_POWER_SEL_ACMP** }

Funciones

- static void SYSCON_set_pll_control (uint8_t m, uint8_t p)

  Configuracion del registro de control del PLL.
- static uint8_t SYSCON_get_pll_lock_status (void)

  Obtener estado de lock del PLL.
- static void SYSCON_set_oscillator_control (SYSCON_bypass_sel_en bypass, SYSCON_freqrange_sel_en freqrange)

  Configurar el registro de control del sistema del oscilador.
- static void SYSCON_set_watchdog_oscillator_control (uint8_t divsel, SYSCON_watchdog_clkana_sel_en clkana_sel)

  Configuracion del registro de control del oscilador del watchdog.
- static void SYSCON_set_fro_direct (void)

  Configuracion del FRO para que sea el oscilador directo (sin division)
- static void SYSCON_clear_fro_direct (void)

  Configuracion del FRO para que sea el oscilador dividido (con division dependiente de FAIM)

- static void SYSCON_set_pll_clk_source (SYSCON_pll_source_sel_en pll_source)

  Configuracion de la fuente de clock para el PLL.
- static void **SYSCON_set_system_clock_source** (SYSCON_main_clock_sel_en clock_selection)
- static void **SYSCON_set_system_clock_divider** (uint8_t divider)
- static void SYSCON_set_capacitive_clock_source (SYSCON_capacitive_clock_sel_en source_sel)

  Seleccion de la fuente de clock para el periferico de control de touch capacitivo.
- static void SYSCON_set_adc_clock (SYSCON_adc_clock_sel_en source_sel, uint8_t div)

  Seleccion de clock y divisor para el periferico ADC.
- static void SYSCON_set_sct_clock (SYSCON_sct_clock_sel_en source_sel, uint8_t div)

  Seleccion de clock y divisor para el periferico SCT.
- static void SYSCON_set_ext_clock_source (SYSCON_ext_clock_source_sel_en source_selection)

  Seleccion de fuente para el clock externo.
- static void SYSCON_enable_clock (SYSCON_enable_clock_sel_en peripheral)

  Habilitacion del clock de un periferico.
- static void SYSCON_disable_clock (SYSCON_enable_clock_sel_en peripheral)

  Inhabilitacion del clock de un periferico.
- static void SYSCON_assert_reset (SYSCON_reset_sel_en peripheral)

  Generar el reset en el periferico seleccionado.
- static void SYSCON_clear_reset (SYSCON_reset_sel_en peripheral)

  Liberar el reset en el periferico seleccionado.
- static void SYSCON_set_peripheral_clock_source (SYSCON_peripheral_sel_en peripheral, SYSCON_↩ peripheral_clock_sel_en clock)

  Seleccion de fuente de clock para los distintos perifericos.
- static void SYSCON_set_frg_config (uint8_t frg_selection, SYSCON_frg_clock_sel_en clock_source, uint8↩ _t mul, uint8_t div)

  Configuracion del FRG.
- static void SYSCON_set_clkout_config (SYSCON_clkout_source_sel_en clock_source, uint8_t divider)

  Seleccion de fuente para el CLOCK OUT.
- static uint32_t SYSCON_get_por_pio_status_register (uint8_t inst)

  Leer el contenido de PIOPORCAPn.
- static void SYSCON_set_iocon_glitch_divider (uint8_t inst, uint8_t div)

  Fijar el divisor para el filtro de glitches del IOCON.
- static void SYSCON_set_bod_control (SYSCON_bod_level_en reset_level, SYSCON_bod_level_en bod_↩ level, SYSCON_bod_enale_en reset_enable)

  Configurar el registro de control del brown-out detector.
- static uint32_t SYSCON_get_systick_calib (void)

  Obtener el valor de calibracion del SYSTICK.
- static uint8_t SYSCON_get_irq_latency (void)

  Obtener el valor de latencia de interrupciones del MCU.
- static void SYSCON_set_nmi_source (uint8_t irq, SYSCON_nmi_enable_en enable)

  Fijar que numero de interrupcion actuara como NMI.
- static void SYSCON_set_pinint_pin (uint8_t channel, GPIO_portpin_en portpin)

  Configurar pin a utilizar como fuente de PININT.
- static void SYSCON_enable_wakeup_source (SYSCON_enable_wakeup_sel_en peripheral)

  Habilitar alguna de las interrupciones del periferico seleccionado como fuente de wakeup.
- static void SYSCON_disable_wakeup_source (SYSCON_enable_wakeup_sel_en peripheral)

  Inhabilitar alguna de las interrupciones del periferico seleccionado como fuente de wakeup.
- static void SYSCON_deep_sleep_power_bod (SYSCON_deep_sleep_power_en power)

  Habilitar o inhabilitacion de la alimentacion del BOD en deep sleep.
- static void SYSCON_deep_sleep_power_wdtosc (SYSCON_deep_sleep_power_en power)

  Habilitar o inhabilitacion de la alimentacion del WDTOSC en deep sleep.

- static void SYSCON_set_powered_on_wakeup (SYSCON_wakeup_power_sel_en peripheral)
  Fijar que un periferico comience encendido al haber un wakeup.
- static void SYSCON_clear_powered_on_wakeup (SYSCON_wakeup_power_sel_en peripheral)
  Fijar que un periferico comience apagado al haber un wakeup.
- static void SYSCON_power_up_peripheral (SYSCON_power_sel_en peripheral)
  Encender el periferico seleccionado.
- static void SYSCON_power_down_peripheral (SYSCON_power_sel_en peripheral)
  Apagar el periferico seleccionado.
- static uint32_t SYSCON_get_device_id (void)
  Obtener el Device ID.

Variables

- volatile SYSCON_per_t ∗const SYSCON
  Periferico SYSCON.

7.23.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del SYSCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.23.2. Documentación de las funciones

7.23.2.1. SYSCON_set_pll_control()

```
static void SYSCON_set_pll_control (
            uint8_t m,
            uint8_t p )  [inline], [static]
```

Configuracion del registro de control del PLL.

Parámetros

in	*m*	Valor del divisor del feedback
in	*p*	Valor del post divisor

7.23.2.2. SYSCON_get_pll_lock_status()

```
static uint8_t SYSCON_get_pll_lock_status (
            void  )  [inline], [static]
```

Obtener estado de lock del PLL.

Devuelve

Valor del estado de lock del PLL

7.23.2.3. SYSCON_set_oscillator_control()

```
static void SYSCON_set_oscillator_control (
            SYSCON_bypass_sel_en bypass,
            SYSCON_freqrange_sel_en freqrange )  [inline], [static]
```

Configurar el registro de control del sistema del oscilador.

Parámetros

in	*bypass*	Seleccion de bypass
in	*freqrange*	Rango de frecuencia a utilizar

7.23.2.4. SYSCON_set_watchdog_oscillator_control()

```
static void SYSCON_set_watchdog_oscillator_control (
            uint8_t divsel,
            SYSCON_watchdog_clkana_sel_en clkana_sel )  [inline], [static]
```

Configuracion del registro de control del oscilador del watchdog.

Parámetros

in	*divsel*	Seleccion de divsel
in	*clkana_sel*	Seleccion de frecuencia base

7.23.2.5. SYSCON_set_pll_clk_source()

```
static void SYSCON_set_pll_clk_source (
```

```
            SYSCON_pll_source_sel_en pll_source )  [inline], [static]
```

Configuracion de la fuente de clock para el PLL.

Parámetros

in	*pll_source*	Fuente de entrada para el PLL

7.23.2.6. SYSCON_set_capacitive_clock_source()

```
static void SYSCON_set_capacitive_clock_source (
            SYSCON_capacitive_clock_sel_en source_sel )  [inline], [static]
```

Seleccion de la fuente de clock para el periferico de control de touch capacitivo.

Parámetros

in	*source_sel*	Seleccion de fuente de clock deseada

7.23.2.7. SYSCON_set_adc_clock()

```
static void SYSCON_set_adc_clock (
            SYSCON_adc_clock_sel_en source_sel,
            uint8_t div )  [inline], [static]
```

Seleccion de clock y divisor para el periferico ADC.

Parámetros

in	*clock_sel*	Fuente de clock deseada para el periferico
in	*div*	Divisor deseado

7.23.2.8. SYSCON_set_sct_clock()

```
static void SYSCON_set_sct_clock (
            SYSCON_sct_clock_sel_en source_sel,
            uint8_t div )  [inline], [static]
```

Seleccion de clock y divisor para el periferico SCT.

Parámetros

in	*clock_sel*	Fuente de clock deseada para el periferico
in	*div*	Divisor deseado

7.23.2.9. SYSCON_set_ext_clock_source()

```
static void SYSCON_set_ext_clock_source (
            SYSCON_ext_clock_source_sel_en source_selection )  [inline], [static]
```

Seleccion de fuente para el clock externo.

Parámetros

in	*source_selection*	Seleccion deseada

7.23.2.10. SYSCON_enable_clock()

```
static void SYSCON_enable_clock (
            SYSCON_enable_clock_sel_en peripheral )  [inline], [static]
```

Habilitacion del clock de un periferico.

Parámetros

in	*peripheral*	Periferico en el cual habilitar el clock

7.23.2.11. SYSCON_disable_clock()

```
static void SYSCON_disable_clock (
            SYSCON_enable_clock_sel_en peripheral )  [inline], [static]
```

Inhabilitacion del clock de un periferico.

Parámetros

in	*peripheral*	Periferico en el cual habilitar el clock

7.23.2.12. SYSCON_assert_reset()

```
static void SYSCON_assert_reset (
            SYSCON_reset_sel_en peripheral )  [inline], [static]
```

Generar el reset en el periferico seleccionado.

Parámetros

in	*peripheral*	Periferico a generar el reset

7.23.2.13. SYSCON_clear_reset()

```
static void SYSCON_clear_reset (
            SYSCON_reset_sel_en peripheral )  [inline], [static]
```

Liberar el reset en el periferico seleccionado.

Parámetros

in	*peripheral*	Periferico a liberar el reset

7.23.2.14. SYSCON_set_peripheral_clock_source()

```
static void SYSCON_set_peripheral_clock_source (
            SYSCON_peripheral_sel_en peripheral,
            SYSCON_peripheral_clock_sel_en clock )  [inline], [static]
```

Seleccion de fuente de clock para los distintos perifericos.

Parámetros

in	*peripheral*	Periferico cuya fuente seleccionar
in	*clock*	Fuente de clock para el periferico seleccionada

7.23.2.15. SYSCON_set_frg_config()

```
static void SYSCON_set_frg_config (
            uint8_t frg_selection,
            SYSCON_frg_clock_sel_en clock_source,
            uint8_t mul,
            uint8_t div )  [inline], [static]
```

Configuracion del FRG.

Parámetros

in	*frg_selection*	Cual de los FRG configurar, cero o uno
in	*clock_source*	Fuente de clock del FRG
in	*mul*	Multiplicador del FRG 0 $\sim$ 255
in	*div*	Divisor del FRG 0$\sim$255

7.23.2.16. SYSCON_set_clkout_config()

```
static void SYSCON_set_clkout_config (
            SYSCON_clkout_source_sel_en clock_source,
            uint8_t divider )  [inline], [static]
```

Seleccion de fuente para el CLOCK OUT.

Parámetros

in	*clock_source*	Fuente deseada
in	*divider*	Divisor del CLOCK OUT

7.23.2.17. SYSCON_get_por_pio_status_register()

```
static uint32_t SYSCON_get_por_pio_status_register (
            uint8_t inst )  [inline], [static]
```

Leer el contenido de PIOPORCAPn.

Parámetros

in	*inst*	Instancia a leer (0 o 1)

Devuelve

Valor del registro leido

7.23.2.18. SYSCON_set_iocon_glitch_divider()

```
static void SYSCON_set_iocon_glitch_divider (
            uint8_t inst,
            uint8_t div )  [inline], [static]
```

Fijar el divisor para el filtro de glitches del IOCON.

Parámetros

in	*inst*	Instancia a escribir
in	*div*	Divisor deseado

7.23.2.19. SYSCON_set_bod_control()

```
static void SYSCON_set_bod_control (
             SYSCON_bod_level_en reset_level,
             SYSCON_bod_level_en bod_level,
             SYSCON_bod_enale_en reset_enable )  [inline], [static]
```

Configurar el registro de control del brown-out detector.

Parámetros

in	*reset_level*	Nivel deseado para el reset
in	*bod_level*	Nivel deseado para el BOD
in	*reset_enable*	Habilitacion deseada para el reset

7.23.2.20. SYSCON_get_systick_calib()

```
static uint32_t SYSCON_get_systick_calib (
             void  )  [inline], [static]
```

Obtener el valor de calibracion del SYSTICK.

Devuelve

Valor de calibracion del SYSTICK

7.23.2.21. SYSCON_get_irq_latency()

```
static uint8_t SYSCON_get_irq_latency (
             void  )  [inline], [static]
```

Obtener el valor de latencia de interrupciones del MCU.

Devuelve

Valor de latencia de interrupcion

7.23.2.22. SYSCON_set_nmi_source()

```
static void SYSCON_set_nmi_source (
             uint8_t irq,
             SYSCON_nmi_enable_en enable )  [inline], [static]
```

Fijar que numero de interrupcion actuara como NMI.

Parámetros

in	*irq*	Numero de interrupcion deseado
in	*enable*	Habilitacion/inhabilitacion de la NMI

7.23.2.23. SYSCON_set_pinint_pin()

```
static void SYSCON_set_pinint_pin (
            uint8_t channel,
            GPIO_portpin_en portpin )  [inline], [static]
```

Configurar pin a utilizar como fuente de PININT.

Parámetros

in	*channel*	Canal de PININT a configurar
in	*portpin*	Puerto/pin a utilizar

7.23.2.24. SYSCON_enable_wakeup_source()

```
static void SYSCON_enable_wakeup_source (
            SYSCON_enable_wakeup_sel_en peripheral )  [inline], [static]
```

Habilitar alguna de las interrupciones del periferico seleccionado como fuente de wakeup.

Parámetros

in	*peripheral*	Periferico deseado

7.23.2.25. SYSCON_disable_wakeup_source()

```
static void SYSCON_disable_wakeup_source (
            SYSCON_enable_wakeup_sel_en peripheral )  [inline], [static]
```

Inhabilitar alguna de las interrupciones del periferico seleccionado como fuente de wakeup.

Parámetros

in	*peripheral*	Periferico deseado

7.23.2.26. SYSCON_deep_sleep_power_bod()

```
static void SYSCON_deep_sleep_power_bod (
            SYSCON_deep_sleep_power_en power )  [inline], [static]
```

Habilitar o inhabilitacion de la alimentacion del BOD en deep sleep.

Parámetros

in	*power*	Habilitacion o inhabilitacion de la alimentacion

7.23.2.27. SYSCON_deep_sleep_power_wdtosc()

```
static void SYSCON_deep_sleep_power_wdtosc (
            SYSCON_deep_sleep_power_en power )  [inline], [static]
```

Habilitar o inhabilitacion de la alimentacion del WDTOSC en deep sleep.

Parámetros

in	*power*	Habilitacion o inhabilitacion de la alimentacion

7.23.2.28. SYSCON_set_powered_on_wakeup()

```
static void SYSCON_set_powered_on_wakeup (
            SYSCON_wakeup_power_sel_en peripheral )  [inline], [static]
```

Fijar que un periferico comience encendido al haber un wakeup.

Parámetros

in	*peripheral*	Periferico que comenzara encendido al haber un wakeup

7.23.2.29. SYSCON_clear_powered_on_wakeup()

```
static void SYSCON_clear_powered_on_wakeup (
            SYSCON_wakeup_power_sel_en peripheral )  [inline], [static]
```

Fijar que un periferico comience apagado al haber un wakeup.

Parámetros

in	*peripheral*	Periferico que comenzara apagado al haber un wakeup

7.23.2.30. SYSCON_power_up_peripheral()

```
static void SYSCON_power_up_peripheral (
            SYSCON_power_sel_en peripheral )  [inline], [static]
```

Encender el periferico seleccionado.

Parámetros

in	*peripheral*	Periferico a encender

7.23.2.31. SYSCON_power_down_peripheral()

```
static void SYSCON_power_down_peripheral (
            SYSCON_power_sel_en peripheral )  [inline], [static]
```

Apagar el periferico seleccionado.

Parámetros

in	*peripheral*	Periferico a encender

7.23.2.32. SYSCON_get_device_id()

```
static uint32_t SYSCON_get_device_id (
            void  )  [inline], [static]
```

Obtener el Device ID.

Devuelve

Device ID

7.24. Referencia del Archivo includes/hpl/HPL_SYSTICK.h

Declaraciones a nivel de abstraccion de periferico del SYSTICK (LPC845)

```
#include <HRI_SYSTICK.h>
```

Dependencia gráfica adjunta para HPL_SYSTICK.h:

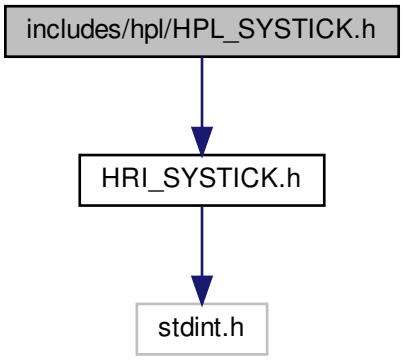
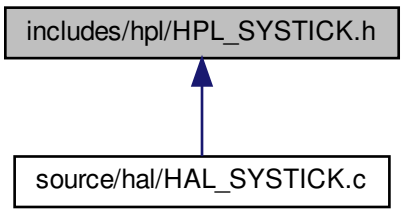


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **SYSTICK_clock_source_en** { **SYSTICK_CLOCK_SOURCE_MAIN_CLOCK_2** = 0, **SYSTICK_CL↩ OCK_SOURCE_MAIN_CLOCK** }

Funciones

- static void **SYSTICK_enable_count** (void)
- static void **SYSTICK_disable_count** (void)
- static void **SYSTICK_enable_interrupt** (void)

- static void **SYSTICK_disable_interrupt** (void)
- static void SYSTICK_select_clock_source (SYSTICK_clock_source_en clock_source)

  Seleccion de fuente de clock.
- static uint8_t SYSTICK_get_count_flag (void)

  Obtener flag de conteo terminado.
- static void SYSTICK_set_reload (uint32_t reload)

  Fijar el valor de reload.
- static void SYSTICK_set_clear_current_value (void)

  Limpiar el conteo actual.

Variables

- volatile SYSTICK_reg_t ∗const SYSTICK

  Periferico SYSTICK.

7.24.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del SYSTICK (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.24.2. Documentación de las funciones

7.24.2.1. SYSTICK_select_clock_source()

```
static void SYSTICK_select_clock_source (
            SYSTICK_clock_source_en clock_source )  [inline], [static]
```

Seleccion de fuente de clock.

Parámetros

in	*clock_source*	Fuente deseada

7.24.2.2. SYSTICK_get_count_flag()

```
static uint8_t SYSTICK_get_count_flag (
            void  )  [inline], [static]
```

Obtener flag de conteo terminado.

Devuelve

Si el SYSTICK habia terminado la cuenta antes de leer el registro, devuelve 1

7.25. Referencia del Archivo includes/hpl/HPL_UART.h

Declaraciones a nivel de abstraccion de periferico del UART (LPC845)

```
#include <HRI_UART.h>
```

Dependencia gráfica adjunta para HPL_UART.h:

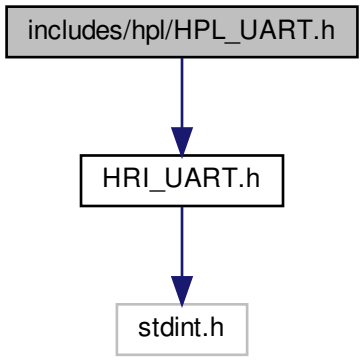
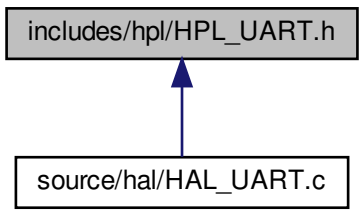


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **UART_datalen_en** { **UART_DATALEN_7BIT** = 0, **UART_DATALEN_8BIT**, **UART_DATALEN_9BIT** }
- enum **UART_parity_en** { **UART_PARITY_NO_PARITY** = 0, **UART_PARITY_EVEN** = 2, **UART_PARITY_ODD** }
- enum **UART_stoplen_en** { **UART_STOPLEN_1BIT** = 0, **UART_STOPLEN_2BIT** }
- enum **UART_sync_mode_en** { **UART_SYNC_MODE_ASYNCHRONOUS** = 0, **UART_SYNC_MODE_SYNCHRONOUS** }
- enum **UART_polarity_en** { **UART_POLARITY_FALLING_EDGE** = 0, **UART_POLARITY_RISING_EDGE** }
- enum **UART_master_mode_en** { **UART_MASTER_MODE_SLAVE** = 0, **UART_MASTER_MODE_MASTER** }
- enum **UART_output_enable_pol_en** { **UART_OUTPUT_ENABLE_POL_LOW** = 0, **UART_OUTPUT_ENABLE_POL_HIGH** }

Funciones

- static void UART_enable (uint8_t inst)

  Habilitacion de una instancia de UART.
- static void UART_disable (uint8_t inst)

  Inhabilitacion de una instancia de UART.
- static void UART_config_data_length (uint8_t inst, UART_datalen_en datalen)

  Configurar largo de palabra.
- static void UART_config_parity (uint8_t inst, UART_parity_en parity)

  Configurar paridad.
- static void UART_config_stop_bits (uint8_t inst, UART_stoplen_en stop_bits)

  Configurar bits de stop.
- static void UART_enable_CTS (uint8_t inst)

  Habilitar CTS.
- static void UART_disable_CTS (uint8_t inst)

  Inhabilitar CTS.
- static void UART_config_sync_mode (uint8_t inst, UART_sync_mode_en sync_mode)

  Configurar modo asincronico/sincronico.
- static void UART_config_clock_polarity (uint8_t inst, UART_polarity_en polarity)

  Configurar polaridad de clock y sampleo (modo sincronico)
- static void UART_config_master_mode (uint8_t inst, UART_master_mode_en master_mode)

  Configurar modo master o slave (modo sincronico)
- static void UART_enable_loopback (uint8_t inst)

  Habilitar modo loopback.
- static void UART_disable_loopback (uint8_t inst)

  Inhabilitar modo loopback.
- static void UART_enable_OETA (uint8_t inst)

  Habilitar turnarround para RS-485.
- static void UART_disable_OETA (uint8_t inst)

  Inhabilitar turnarround para RS-485.
- static void UART_enable_auto_address (uint8_t inst)

  Habilitar auto address.
- static void UART_disable_auto_address (uint8_t inst)

  Inhabilitar auto address.
- static void UART_enable_OESEL (uint8_t inst)

  Habilitar output enable para RS-485.

- static void UART_disable_OESEL (uint8_t inst)

  Inhabilitar output enable para RS-485.
- static void UART_config_OEPOL (uint8_t inst, UART_output_enable_pol_en polarity)

  Configurar polaridad de output enable para RS-485.
- static void UART_enable_rx_invert (uint8_t inst)

  Habilitar inversion para recepcion.
- static void UART_disable_rx_invert (uint8_t inst)

  Inhabilitar inversion para recepcion.
- static void UART_enable_tx_invert (uint8_t inst)

  Habilitar inversion para transmision.
- static void UART_disable_tx_invert (uint8_t inst)

  Inhabilitar inversion para transmision.
- static void UART_assert_break (uint8_t inst)

  Fijar condicion de break.
- static void UART_clear_break (uint8_t inst)

  Liberar condicion de break.
- static void UART_enable_address_detect (uint8_t inst)

  Habilitar address detect.
- static void UART_disable_address_detect (uint8_t inst)

  Inhabilitar address detect.
- static void UART_enable_tx (uint8_t inst)

  Habilitar TX.
- static void UART_disable_tx (uint8_t inst)

  Inhabilitar TX.
- static void UART_enable_continuous_clock (uint8_t inst)

  Habilitar clock continuo (modo sincronico)
- static void UART_disable_continuous_clock (uint8_t inst)

  Inhabilitar clock continuo (modo sincronico)
- static void UART_enable_autoclear_continuous_clock (uint8_t inst)

  Habilitar parada de clock continuo en rx (modo sincronico)
- static void UART_disable_autoclear_continuous_clock (uint8_t inst)

  Inhabilitar parada de clock continuo en rx (modo sincronico)
- static void UART_enable_autobaud (uint8_t inst)

  Habilitar auto baud.
- static void UART_disable_autobaud (uint8_t inst)

  Inhabilitar auto baud.
- static uint8_t UART_get_flag_RXRDY (uint8_t inst)

  Obtener estado del flag RXRDY.
- static uint8_t UART_get_flag_RXIDLE (uint8_t inst)

  Obtener estado del flag RXIDLE.
- static uint8_t UART_get_flag_TXRDY (uint8_t inst)

  Obtener estado del flag TXRDY.
- static uint8_t UART_get_flag_TXIDLE (uint8_t inst)

  Obtener estado del flag TXIDLE.
- static uint8_t UART_get_flag_CTS (uint8_t inst)

  Obtener estado del flag CTS.
- static uint8_t UART_get_flag_DELTACTS (uint8_t inst)

  Obtener estado del flag DELTACTS.
- static uint8_t UART_get_flag_TXDISSTAT (uint8_t inst)

  Obtener estado del flag TXDISSTAT.
- static uint8_t UART_get_flag_OVERRUNINT (uint8_t inst)

Obtener estado del flag OVERRUNINT.

- static uint8_t UART_get_flag_RXBRK (uint8_t inst)

  Obtener estado del flag RXBRK.
- static uint8_t UART_get_flag_DELTARXBRK (uint8_t inst)

  Obtener estado del flag DELTARXBRK.
- static uint8_t UART_get_flag_START (uint8_t inst)

  Obtener estado del flag START.
- static uint8_t UART_get_flag_FRAMERRINT (uint8_t inst)

  Obtener estado del flag FRAMERRINT.
- static uint8_t UART_get_flag_PARITYERRINT (uint8_t inst)

  Obtener estado del flag PARITYERRINT.
- static uint8_t UART_get_flag_RXNOISEINT (uint8_t inst)

  Obtener estado del flag RXNOISEINT.
- static uint8_t UART_get_flag_ABERR (uint8_t inst)

  Obtener estado del flag ABERR.
- static void UART_enable_irq_RXRDY (uint8_t inst)

  Habilitar interrupcion en RXRDY.
- static void UART_enable_irq_TXRDY (uint8_t inst)

  Habilitar interrupcion en TXRDY.
- static void UART_enable_irq_TXIDLE (uint8_t inst)

  Habilitar interrupcion en TXIDLE.
- static void UART_enable_irq_DELTACTS (uint8_t inst)

  Habilitar interrupcion en DELTACTS.
- static void UART_enable_irq_TXDISEN (uint8_t inst)

  Habilitar interrupcion en TXDISEN.
- static void UART_enable_irq_OVERRUN (uint8_t inst)

  Habilitar interrupcion en OVERRUN.
- static void UART_enable_irq_DELTARXBRK (uint8_t inst)

  Habilitar interrupcion en DELTARXBRK.
- static void UART_enable_irq_START (uint8_t inst)

  Habilitar interrupcion en START.
- static void UART_enable_irq_FRAMERR (uint8_t inst)

  Habilitar interrupcion en FRAMERR.
- static void UART_enable_irq_PARITYERR (uint8_t inst)

  Habilitar interrupcion en PARITYERR.
- static void UART_enable_irq_RXNOISE (uint8_t inst)

  Habilitar interrupcion en RXNOISE.
- static void UART_enable_irq_ABERR (uint8_t inst)

  Habilitar interrupcion en ABERR.
- static void UART_disable_irq_RXRDY (uint8_t inst)

  Inhabilitar interrupcion en RXRDY.
- static void UART_disable_irq_TXRDY (uint8_t inst)

  Inhabilitar interrupcion en TXRDY.
- static void UART_disable_irq_TXIDLE (uint8_t inst)

  Inhabilitar interrupcion en TXIDLE.
- static void UART_disable_irq_DELTACTS (uint8_t inst)

  Inhabilitar interrupcion en DELTACTS.
- static void UART_disable_irq_TXDISEN (uint8_t inst)

  Inhabilitar interrupcion en TXDISEN.
- static void UART_disable_irq_OVERRUN (uint8_t inst)

  Inhabilitar interrupcion en OVERRUN.

- static void UART_disable_irq_DELTARXBRK (uint8_t inst)

  Inhabilitar interrupcion en DELTARXBRK.
- static void UART_disable_irq_START (uint8_t inst)

  Inhabilitar interrupcion en START.
- static void UART_disable_irq_FRAMERR (uint8_t inst)

  Inhabilitar interrupcion en FRAMERR.
- static void UART_disable_irq_PARITYERR (uint8_t inst)

  Inhabilitar interrupcion en PARITYERR.
- static void UART_disable_irq_RXNOISE (uint8_t inst)

  Inhabilitar interrupcion en RXNOISE.
- static void UART_disable_irq_ABERR (uint8_t inst)

  Inhabilitar interrupcion en ABERR.
- static uint32_t UART_get_data (uint8_t inst)

  Obtener ultimo dato recibido.
- static uint32_t UART_get_data_and_status (uint8_t inst, uint8_t *frame, uint8_t *parity, uint8_t *noise)

  Obtener ultimo dato recibido con flags de errores.
- static void UART_write_data (uint8_t inst, uint32_t data)

  Iniciar transmision de dato.
- static void UART_set_BRGVAL (uint8_t inst, uint32_t brg)

  Escribir el registro BRG.
- static uint8_t UART_get_irq_status_RXRDY (uint8_t inst)

  Leer estado de interrupcion RXRDY.
- static uint8_t UART_get_irq_status_TXRDY (uint8_t inst)

  Leer estado de interrupcion TXRDY.
- static uint8_t UART_get_irq_status_TXIDLE (uint8_t inst)

  Leer estado de interrupcion TXIDLE.
- static uint8_t UART_get_irq_status_DELTACTS (uint8_t inst)

  Leer estado de interrupcion DELTACTS.
- static uint8_t UART_get_irq_status_TXDIS (uint8_t inst)

  Leer estado de interrupcion TXDIS.
- static uint8_t UART_get_irq_status_OVERRUN (uint8_t inst)

  Leer estado de interrupcion OVERRUN.
- static uint8_t UART_get_irq_status_DELTARXBRK (uint8_t inst)

  Leer estado de interrupcion DELTARXBRK.
- static uint8_t UART_get_irq_status_START (uint8_t inst)

  Leer estado de interrupcion START.
- static uint8_t UART_get_irq_status_FRAMERR (uint8_t inst)

  Leer estado de interrupcion FRAMERR.
- static uint8_t UART_get_irq_status_PARITYERR (uint8_t inst)

  Leer estado de interrupcion PARITYERR.
- static uint8_t UART_get_irq_status_RXNOISE (uint8_t inst)

  Leer estado de interrupcion RXNOISE.
- static uint8_t UART_get_irq_status_ABERR (uint8_t inst)

  Leer estado de interrupcion ABERR.
- static void UART_set_OSRVAL (uint8_t inst, uint32_t osr)

  Escribir el registro OSR.
- static void UART_set_address (uint8_t inst, uint32_t addr)

  Escribir el registro ADDR.

Variables

- volatile UART_per_t *const UART [ ]
  Perifericos USART.

7.25.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del UART (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.25.2. Documentación de las funciones

7.25.2.1. UART_enable()

```
static void UART_enable (
            uint8_t inst )  [inline], [static]
```

Habilitacion de una instancia de UART.

Parámetros

in	*inst*	Instancia a habilitar

7.25.2.2. UART_disable()

```
static void UART_disable (
            uint8_t inst )  [inline], [static]
```

Inhabilitacion de una instancia de UART.

Parámetros

in	*inst*	Instancia a inhabilitar

7.25.2.3. UART_config_data_length()

```
static void UART_config_data_length (
            uint8_t inst,
            UART_datalen_en datalen )  [inline], [static]
```

Configurar largo de palabra.

Parámetros

in	*inst*	Instancia a configurar
in	*datalen*	Seleccion de largo de palabra deseado

7.25.2.4. UART_config_parity()

```
static void UART_config_parity (
            uint8_t inst,
            UART_parity_en parity )  [inline], [static]
```

Configurar paridad.

Parámetros

in	*inst*	Instancia a configurar
in	*parity*	Seleccion de paridad deseada

7.25.2.5. UART_config_stop_bits()

```
static void UART_config_stop_bits (
            uint8_t inst,
            UART_stoplen_en stop_bits )  [inline], [static]
```

Configurar bits de stop.

Parámetros

in	*inst*	Instancia a configurar
in	*stop_bits*	Seleccion de bits de stop deseados

7.25.2.6. UART_enable_CTS()

```
static void UART_enable_CTS (
            uint8_t inst )  [inline], [static]
```

Habilitar CTS.

Parámetros

in	*inst*	Instancia a habilitar

7.25.2.7. UART_disable_CTS()

```
static void UART_disable_CTS (
            uint8_t inst )  [inline], [static]
```

Inhabilitar CTS.

Parámetros

in	*inst*	Instancia a inhabilitar

7.25.2.8. UART_config_sync_mode()

```
static void UART_config_sync_mode (
            uint8_t inst,
            UART_sync_mode_en sync_mode )  [inline], [static]
```

Configurar modo asincronico/sincronico.

Parámetros

in	*inst*	Instancia a configurar
in	*sync_mode*	Modo desesado

7.25.2.9. UART_config_clock_polarity()

```
static void UART_config_clock_polarity (
            uint8_t inst,
            UART_polarity_en polarity )  [inline], [static]
```

Configurar polaridad de clock y sampleo (modo sincronico)

Parámetros

in	*inst*	Instancia a configurar
in	*polarity*	Polaridad y sampleo desesado

7.25.2.10. UART_config_master_mode()

```
static void UART_config_master_mode (
            uint8_t inst,
            UART_master_mode_en master_mode )  [inline], [static]
```

Configurar modo master o slave (modo sincronico)

Parámetros

in	*inst*	Instancia a configurar
in	*master_mode*	Modo deseado

7.25.2.11. UART_enable_loopback()

```
static void UART_enable_loopback (
            uint8_t inst )  [inline], [static]
```

Habilitar modo loopback.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.12. UART_disable_loopback()

```
static void UART_disable_loopback (
            uint8_t inst )  [inline], [static]
```

Inhabilitar modo loopback.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.13. UART_enable_OETA()

```
static void UART_enable_OETA (
            uint8_t inst )  [inline], [static]
```

Habilitar turnarround para RS-485.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.14. UART_disable_OETA()

```
static void UART_disable_OETA (
            uint8_t inst )  [inline], [static]
```

Inhabilitar turnarround para RS-485.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.15. UART_enable_auto_address()

```
static void UART_enable_auto_address (
            uint8_t inst )  [inline], [static]
```

Habilitar auto address.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.16. UART_disable_auto_address()

```
static void UART_disable_auto_address (
            uint8_t inst )  [inline], [static]
```

Inhabilitar auto address.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.17. UART_enable_OESEL()

```
static void UART_enable_OESEL (
            uint8_t inst )  [inline], [static]
```

Habilitar output enable para RS-485.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.18. UART_disable_OESEL()

```
static void UART_disable_OESEL (
            uint8_t inst )  [inline], [static]
```

Inhabilitar output enable para RS-485.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.19. UART_config_OEPOL()

```
static void UART_config_OEPOL (
            uint8_t inst,
            UART_output_enable_pol_en polarity )  [inline], [static]
```

Configurar polaridad de output enable para RS-485.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.20. UART_enable_rx_invert()

```
static void UART_enable_rx_invert (
            uint8_t inst )  [inline], [static]
```

Habilitar inversion para recepcion.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.21. UART_disable_rx_invert()

```
static void UART_disable_rx_invert (
            uint8_t inst )  [inline], [static]
```

Inhabilitar inversion para recepcion.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.22. UART_enable_tx_invert()

```
static void UART_enable_tx_invert (
            uint8_t inst )  [inline], [static]
```

Habilitar inversion para transmision.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.23. UART_disable_tx_invert()

```
static void UART_disable_tx_invert (
            uint8_t inst )  [inline], [static]
```

Inhabilitar inversion para transmision.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.24. UART_assert_break()

```
static void UART_assert_break (
            uint8_t inst )  [inline], [static]
```

Fijar condicion de break.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.25. UART_clear_break()

```
static void UART_clear_break (
            uint8_t inst )  [inline], [static]
```

Liberar condicion de break.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.26. UART_enable_address_detect()

```
static void UART_enable_address_detect (
            uint8_t inst )  [inline], [static]
```

Habilitar address detect.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.27. UART_disable_address_detect()

```
static void UART_disable_address_detect (
            uint8_t inst )  [inline], [static]
```

Inhabilitar address detect.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.28. UART_enable_tx()

```
static void UART_enable_tx (
            uint8_t inst )  [inline], [static]
```

Habilitar TX.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.29. UART_disable_tx()

```
static void UART_disable_tx (
            uint8_t inst )  [inline], [static]
```

Inhabilitar TX.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.30. UART_enable_continuous_clock()

```
static void UART_enable_continuous_clock (
            uint8_t inst )  [inline], [static]
```

Habilitar clock continuo (modo sincronico)

Parámetros

in	*inst*	Instancia a configurar

7.25.2.31. UART_disable_continuous_clock()

```
static void UART_disable_continuous_clock (
            uint8_t inst )  [inline], [static]
```

Inhabilitar clock continuo (modo sincronico)

Parámetros

in	*inst*	Instancia a configurar

7.25.2.32. UART_enable_autoclear_continuous_clock()

```
static void UART_enable_autoclear_continuous_clock (
            uint8_t inst )  [inline], [static]
```

Habilitar parada de clock continuo en rx (modo sincronico)

Parámetros

in	*inst*	Instancia a configurar

7.25.2.33. UART_disable_autoclear_continuous_clock()

```
static void UART_disable_autoclear_continuous_clock (
            uint8_t inst )  [inline], [static]
```

Inhabilitar parada de clock continuo en rx (modo sincronico)

Parámetros

in	*inst*	Instancia a configurar

7.25.2.34. UART_enable_autobaud()

```
static void UART_enable_autobaud (
            uint8_t inst )  [inline], [static]
```

Habilitar auto baud.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.35. UART_disable_autobaud()

```
static void UART_disable_autobaud (
            uint8_t inst )  [inline], [static]
```

Inhabilitar auto baud.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.36. UART_get_flag_RXRDY()

```
static uint8_t UART_get_flag_RXRDY (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag RXRDY.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.37. UART_get_flag_RXIDLE()

```
static uint8_t UART_get_flag_RXIDLE (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag RXIDLE.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.38. UART_get_flag_TXRDY()

```
static uint8_t UART_get_flag_TXRDY (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag TXRDY.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.39. UART_get_flag_TXIDLE()

```
static uint8_t UART_get_flag_TXIDLE (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag TXIDLE.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.40. UART_get_flag_CTS()

```
static uint8_t UART_get_flag_CTS (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag CTS.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.41. UART_get_flag_DELTACTS()

```
static uint8_t UART_get_flag_DELTACTS (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag DELTACTS.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.42. UART_get_flag_TXDISSTAT()

```
static uint8_t UART_get_flag_TXDISSTAT (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag TXDISSTAT.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.43. UART_get_flag_OVERRUNINT()

```
static uint8_t UART_get_flag_OVERRUNINT (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag OVERRUNINT.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.44. UART_get_flag_RXBRK()

```
static uint8_t UART_get_flag_RXBRK (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag RXBRK.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.45. UART_get_flag_DELTARXBRK()

```
static uint8_t UART_get_flag_DELTARXBRK (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag DELTARXBRK.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.46. UART_get_flag_START()

```
static uint8_t UART_get_flag_START (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag START.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.47. UART_get_flag_FRAMERRINT()

```
static uint8_t UART_get_flag_FRAMERRINT (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag FRAMERRINT.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.48. UART_get_flag_PARITYERRINT()

```
static uint8_t UART_get_flag_PARITYERRINT (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag PARITYERRINT.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.49. UART_get_flag_RXNOISEINT()

```
static uint8_t UART_get_flag_RXNOISEINT (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag RXNOISEINT.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.50. UART_get_flag_ABERR()

```
static uint8_t UART_get_flag_ABERR (
            uint8_t inst )  [inline], [static]
```

Obtener estado del flag ABERR.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Estado del flag

7.25.2.51. UART_enable_irq_RXRDY()

```
static void UART_enable_irq_RXRDY (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en RXRDY.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.52. UART_enable_irq_TXRDY()

```
static void UART_enable_irq_TXRDY (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en TXRDY.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.53. UART_enable_irq_TXIDLE()

```
static void UART_enable_irq_TXIDLE (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en TXIDLE.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.54. UART_enable_irq_DELTACTS()

```
static void UART_enable_irq_DELTACTS (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en DELTACTS.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.55. UART_enable_irq_TXDISEN()

```
static void UART_enable_irq_TXDISEN (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en TXDISEN.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.56. UART_enable_irq_OVERRUN()

```
static void UART_enable_irq_OVERRUN (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en OVERRUN.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.57. UART_enable_irq_DELTARXBRK()

```
static void UART_enable_irq_DELTARXBRK (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en DELTARXBRK.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.58. UART_enable_irq_START()

```
static void UART_enable_irq_START (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en START.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.59. UART_enable_irq_FRAMERR()

```
static void UART_enable_irq_FRAMERR (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en FRAMERR.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.60. UART_enable_irq_PARITYERR()

```
static void UART_enable_irq_PARITYERR (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en PARITYERR.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.61. UART_enable_irq_RXNOISE()

```
static void UART_enable_irq_RXNOISE (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en RXNOISE.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.62. UART_enable_irq_ABERR()

```
static void UART_enable_irq_ABERR (
            uint8_t inst )  [inline], [static]
```

Habilitar interrupcion en ABERR.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.63. UART_disable_irq_RXRDY()

```
static void UART_disable_irq_RXRDY (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en RXRDY.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.64. UART_disable_irq_TXRDY()

```
static void UART_disable_irq_TXRDY (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en TXRDY.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.65. UART_disable_irq_TXIDLE()

```
static void UART_disable_irq_TXIDLE (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en TXIDLE.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.66. UART_disable_irq_DELTACTS()

```
static void UART_disable_irq_DELTACTS (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en DELTACTS.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.67. UART_disable_irq_TXDISEN()

```
static void UART_disable_irq_TXDISEN (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en TXDISEN.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.68. UART_disable_irq_OVERRUN()

```
static void UART_disable_irq_OVERRUN (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en OVERRUN.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.69. UART_disable_irq_DELTARXBRK()

```
static void UART_disable_irq_DELTARXBRK (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en DELTARXBRK.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.70. UART_disable_irq_START()

```
static void UART_disable_irq_START (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en START.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.71. UART_disable_irq_FRAMERR()

```
static void UART_disable_irq_FRAMERR (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en FRAMERR.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.72. UART_disable_irq_PARITYERR()

```
static void UART_disable_irq_PARITYERR (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en PARITYERR.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.73. UART_disable_irq_RXNOISE()

```
static void UART_disable_irq_RXNOISE (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en RXNOISE.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.74. UART_disable_irq_ABERR()

```
static void UART_disable_irq_ABERR (
            uint8_t inst )  [inline], [static]
```

Inhabilitar interrupcion en ABERR.

Parámetros

in	*inst*	Instancia a configurar

7.25.2.75. UART_get_data()

```
static uint32_t UART_get_data (
            uint8_t inst )  [inline], [static]
```

Obtener ultimo dato recibido.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Ultimo dato recibido

7.25.2.76. UART_get_data_and_status()

```
static uint32_t UART_get_data_and_status (
            uint8_t inst,
            uint8_t * frame,
            uint8_t * parity,
            uint8_t * noise )  [inline], [static]
```

Obtener ultimo dato recibido con flags de errores.

Parámetros

in	*inst*	Instancia a consultar
out	*frame*	Flag correspondiente a frame error
out	*parity*	Flag correspondiente a parity error
out	*noise*	Flag correspondiente a noise error

Devuelve

Ultimo dato recibido

7.25.2.77. UART_write_data()

```
static void UART_write_data (
            uint8_t inst,
            uint32_t data )  [inline], [static]
```

Iniciar transmision de dato.

Parámetros

in	*inst*	Instancia a utilizar
in	*data*	Dato a transmitir

7.25.2.78. UART_set_BRGVAL()

```
static void UART_set_BRGVAL (
            uint8_t inst,
            uint32_t brg )  [inline], [static]
```

Escribir el registro BRG.

Parámetros

in	*inst*	Instancia a configurar
in	*brg*	Valor a escribir en el registro

7.25.2.79. UART_get_irq_status_RXRDY()

```
static uint8_t UART_get_irq_status_RXRDY (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion RXRDY.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.80. UART_get_irq_status_TXRDY()

```
static uint8_t UART_get_irq_status_TXRDY (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion TXRDY.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.81. UART_get_irq_status_TXIDLE()

```
static uint8_t UART_get_irq_status_TXIDLE (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion TXIDLE.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.82. UART_get_irq_status_DELTACTS()

```
static uint8_t UART_get_irq_status_DELTACTS (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion DELTACTS.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.83. UART_get_irq_status_TXDIS()

```
static uint8_t UART_get_irq_status_TXDIS (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion TXDIS.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.84. UART_get_irq_status_OVERRUN()

```
static uint8_t UART_get_irq_status_OVERRUN (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion OVERRUN.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.85. UART_get_irq_status_DELTARXBRK()

```
static uint8_t UART_get_irq_status_DELTARXBRK (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion DELTARXBRK.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.86. UART_get_irq_status_START()

```
static uint8_t UART_get_irq_status_START (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion START.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.87. UART_get_irq_status_FRAMERR()

```
static uint8_t UART_get_irq_status_FRAMERR (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion FRAMERR.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.88. UART_get_irq_status_PARITYERR()

```
static uint8_t UART_get_irq_status_PARITYERR (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion PARITYERR.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.89. UART_get_irq_status_RXNOISE()

```
static uint8_t UART_get_irq_status_RXNOISE (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion RXNOISE.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.90. UART_get_irq_status_ABERR()

```
static uint8_t UART_get_irq_status_ABERR (
            uint8_t inst )  [inline], [static]
```

Leer estado de interrupcion ABERR.

Parámetros

in	*inst*	Instancia a consultar

7.25.2.91. UART_set_OSRVAL()

```
static void UART_set_OSRVAL (
            uint8_t inst,
            uint32_t osr )  [inline], [static]
```

Escribir el registro OSR.

Parámetros

in	*inst*	Instancia a configurar
in	*osr*	Valor a escribir en el registro

7.25.2.92. UART_set_address()

```
static void UART_set_address (
            uint8_t inst,
            uint32_t addr )  [inline], [static]
```

Escribir el registro ADDR.

Parámetros

in	*inst*	Instancia a configurar
in	*brg*	Valor a escribir en el registro

7.26. Referencia del Archivo includes/hpl/HPL_WKT.h

Declaraciones a nivel de abstraccion de periferico del WKT (LPC845)

```
#include <HRI_WKT.h>
```
Dependencia gráfica adjunta para HPL_WKT.h:

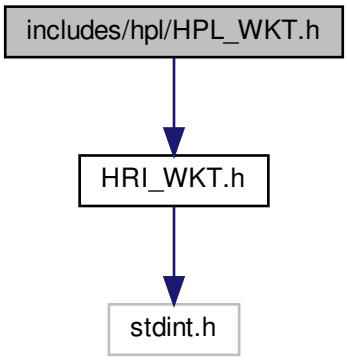
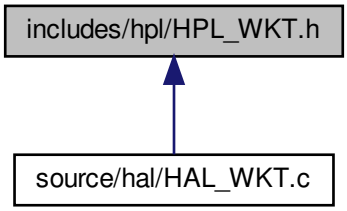


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Enumeraciones

- enum **WKT_clock_source_sel_en** { **WKT_CLOCK_SOURCE_DIVIDED_FRO** = 0, **WKT_CLOCK_SOU↩ RCE_LOW_POWER_CLOCK** }

Funciones

- static void WKT_select_clock_source (WKT_clock_source_sel_en clock_source)

  Seleccionar la fuente de clock para el WKT.
- static uint8_t WKT_get_alarm_flag (void)

  Obtener flag de alarma actual.
- static void WKT_clear_alarm_flag (void)

  Limpiar flag de alarma.
- static void WKT_clear_count (void)

Limpiar el contador del WKT.

- static void WKT_set_internal_clock_source (void)

  Seleccionar fuente de clock interna para el WKT.
- static void WKT_set_external_clock_source (void)

  Seleccionar fuente de clock externa para el WKT.
- static uint32_t WKT_get_current_count (void)

  Obtener cuenta actual del WKT.
- static void WKT_write_count (uint32_t count)

  Fijar la cuenta del WKT.

Variables

- volatile WKT_per_t *const WKT

  Periferico WKT.

7.26.1. Descripción detallada

Declaraciones a nivel de abstraccion de periferico del WKT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.26.2. Documentación de las funciones

7.26.2.1. WKT_select_clock_source()

```
static void WKT_select_clock_source (
            WKT_clock_source_sel_en clock_source )  [inline], [static]
```

Seleccionar la fuente de clock para el WKT.

Parámetros

in	*clock_source*	Fuente de clock deseada

7.26.2.2. WKT_get_alarm_flag()

```
static uint8_t WKT_get_alarm_flag (
             void  )  [inline], [static]
```

Obtener flag de alarma actual.

Devuelve

Estado del flag de alarma actual

7.26.2.3. WKT_get_current_count()

```
static uint32_t WKT_get_current_count (
             void  )  [inline], [static]
```

Obtener cuenta actual del WKT.

Devuelve

Cuenta actual del WKT

7.26.2.4. WKT_write_count()

```
static void WKT_write_count (
             uint32_t count )  [inline], [static]
```

Fijar la cuenta del WKT.

Parámetros

in	*count*	Valor de cuenta deseado

7.27. Referencia del Archivo includes/hri/HRI_ADC.h

Declaraciones a nivel de registros del ADC (LPC845)

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

Dependencia gráfica adjunta para HRI_ADC.h:

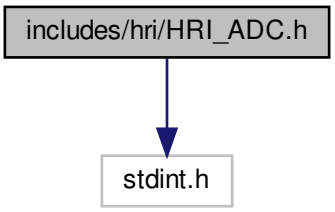
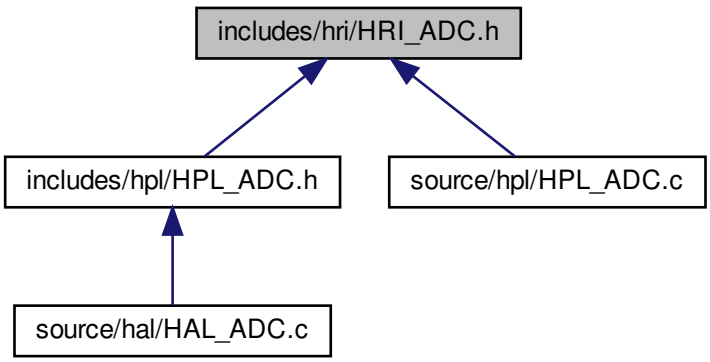


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct ADC_CTRL_reg_t

  Registro de control del ADC. Más...
- struct ADC_SEQ_CTRL_reg_t

  Registro de control de secuencia A y B del ADC. Más...
- struct ADC_SEQ_GDAT_reg_t
- struct ADC_DAT_reg_t
- struct ADC_THR_LOW_reg_t
- struct ADC_THR_HIGH_reg_t
- struct ADC_CHAN_THRSEL_reg_t
- struct ADC_INTEN_reg_t
- struct ADC_FLAGS_reg_t
- struct ADC_TRM_reg_t
- struct ADC_per_t

defines

- #define ADC_BASE 0x4001C000
  Direccion base del ADC.

7.27.1. Descripción detallada

Declaraciones a nivel de registros del ADC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.27.2. Documentación de las estructuras de datos

7.27.2.1. struct ADC_CTRL_reg_t

Registro de control del ADC.

Campos de datos

uint32_t	CLKDIV: 8	Divisor del clock.
uint32_t	ASYNCMODE: 1	0 -> Modo sincronico ; 1 -> Modo asincronico
uint32_t	__pad0__: 1	Reservado.
uint32_t	LPWRMODE: 1	Modo de bajo consumo.
uint32_t	__pad1__: 19	Reservado.
uint32_t	CALMODE: 1	Modo calibracion.
uint32_t	__pad2__: 1	Reservado.

7.27.2.2. struct ADC_SEQ_CTRL_reg_t

Registro de control de secuencia A y B del ADC.

Campos de datos

uint32_t	CHANNELS: 12	Canales habilitados para la conversion en la secuencia.
uint32_t	TRIGGER: 3	Que pin dispara la secuencia.
uint32_t	__pad0__: 3	Reservado.

Campos de datos

uint32_t	TRIGPOL: 1	Polaridad del pin que dispara la secuencia.
uint32_t	SYNCBYPASS: 1	Disparo asincronico/sincronico.
uint32_t	__pad1__: 6	Reservado.
uint32_t	START: 1	Empezar secuencia mediante software.
uint32_t	BURST: 1	Habilitacion de modo rafaga.
uint32_t	SINGLESTEP: 1	Un start activa un paso o la secuencia entera.
uint32_t	LOWPRIO: 1	Prioridad. Solo lo tiene la secuencia A.
uint32_t	MODE: 1	Interrupcion por cada conversion o al final de la secuencia.
uint32_t	SEQ_ENA: 1	Habilitacion de secuencia.

7.27.2.3. struct ADC_SEQ_GDAT_reg_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	RESULT: 12	
uint32_t	THCMPRANGE: 2	
uint32_t	THCMPCROSS: 2	
uint32_t	__pad1__: 6	
uint32_t	CHANNEL: 4	
uint32_t	OVERRUN: 1	
uint32_t	DATAVALID: 1	

7.27.2.4. struct ADC_DAT_reg_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	RESULT: 12	
uint32_t	THCMPRANGE: 2	
uint32_t	THCMPCROSS: 2	
uint32_t	__pad1__: 6	
uint32_t	CHANNEL: 4	
uint32_t	OVERRUN: 1	
uint32_t	DATAVALID: 1	

7.27.2.5. struct ADC_THR_LOW_reg_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	THRLOW: 12	
uint32_t	__pad1__: 16	

7.27.2.6. struct ADC_THR_HIGH_reg_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	THRHIGH: 12	
uint32_t	__pad1__: 16	

7.27.2.7. struct ADC_CHAN_THRSEL_reg_t

Campos de datos

uint32_t	CH0_THRSEL: 1	
uint32_t	CH1_THRSEL: 1	
uint32_t	CH2_THRSEL: 1	
uint32_t	CH3_THRSEL: 1	
uint32_t	CH4_THRSEL: 1	
uint32_t	CH5_THRSEL: 1	
uint32_t	CH6_THRSEL: 1	
uint32_t	CH7_THRSEL: 1	
uint32_t	CH8_THRSEL: 1	
uint32_t	CH9_THRSEL: 1	
uint32_t	CH10_THRSEL: 1	
uint32_t	CH11_THRSEL: 1	
uint32_t	__pad0__: 20	

7.27.2.8. struct ADC_INTEN_reg_t

Campos de datos

uint32_t	SEQA_INTEN: 1	
uint32_t	SEQB_INTEN: 1	
uint32_t	OVR_INTEN: 1	
uint32_t	ADCMPINTEN0: 2	
uint32_t	ADCMPINTEN1: 2	
uint32_t	ADCMPINTEN2: 2	
uint32_t	ADCMPINTEN3: 2	
uint32_t	ADCMPINTEN4: 2	
uint32_t	ADCMPINTEN5: 2	
uint32_t	ADCMPINTEN6: 2	
uint32_t	ADCMPINTEN7: 2	
uint32_t	ADCMPINTEN8: 2	
uint32_t	ADCMPINTEN9: 2	
uint32_t	ADCMPINTEN10: 2	
uint32_t	ADCMPINTEN11: 2	
uint32_t	__pad0__: 5	

7.27.2.9. struct ADC_FLAGS_reg_t

Campos de datos

uint32_t	THCMP0: 1	
uint32_t	THCMP1: 1	
uint32_t	THCMP2: 1	
uint32_t	THCMP3: 1	
uint32_t	THCMP4: 1	
uint32_t	THCMP5: 1	
uint32_t	THCMP6: 1	
uint32_t	THCMP7: 1	
uint32_t	THCMP8: 1	
uint32_t	THCMP9: 1	
uint32_t	THCMP10: 1	
uint32_t	THCMP11: 1	
uint32_t	OVERRUN0: 1	
uint32_t	OVERRUN1: 1	
uint32_t	OVERRUN2: 1	
uint32_t	OVERRUN3: 1	
uint32_t	OVERRUN4: 1	
uint32_t	OVERRUN5: 1	
uint32_t	OVERRUN6: 1	
uint32_t	OVERRUN7: 1	
uint32_t	OVERRUN8: 1	
uint32_t	OVERRUN9: 1	
uint32_t	OVERRUN10: 1	
uint32_t	OVERRUN11: 1	
uint32_t	__pad0__: 2	
uint32_t	SEQA_INT: 1	
uint32_t	SEQB_INT: 1	
uint32_t	THCMP_INT: 1	
uint32_t	OVR_INT: 1	

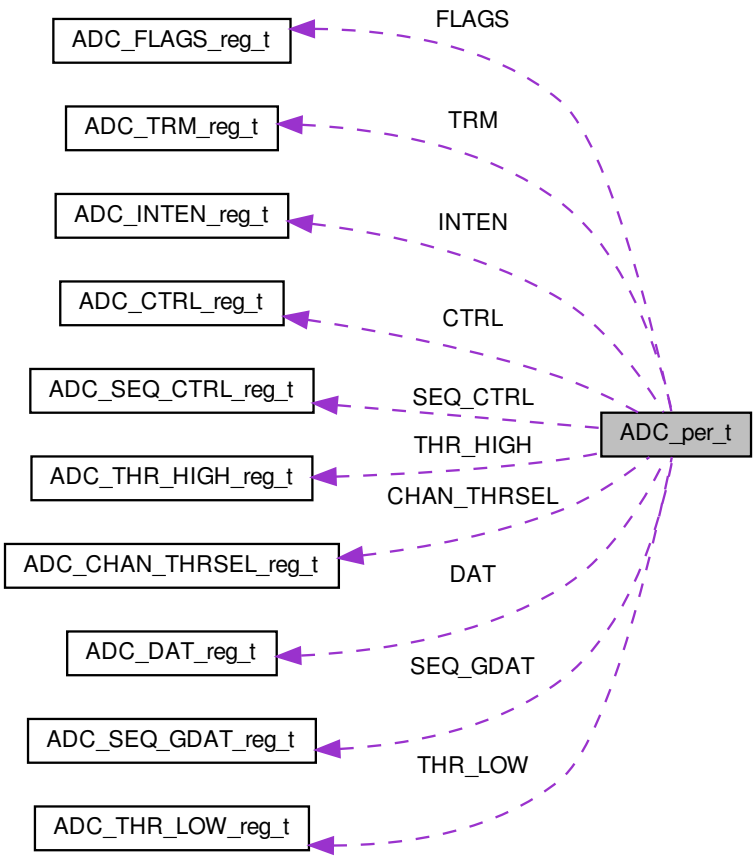
7.27.2.10. struct ADC_TRM_reg_t

Campos de datos

uint32_t	__pad0__: 5	
uint32_t	VRANGE: 1	
uint32_t	__pad1__: 26	

7.27.2.11. struct ADC_per_t

Diagrama de colaboración para ADC_per_t:



Campos de datos

ADC_CTRL_reg_t	CTRL	
const uint32_t	RESERVED_1	
ADC_SEQ_CTRL_reg_t	SEQ_CTRL[2]	
ADC_SEQ_GDAT_reg_t	SEQ_GDAT[2]	
const uint32_t	RESERVED_2[2]	
const ADC_DAT_reg_t	DAT[12]	
ADC_THR_LOW_reg_t	THR_LOW[2]	
ADC_THR_HIGH_reg_t	THR_HIGH[2]	
ADC_CHAN_THRSEL_reg_t	CHAN_THRSEL	
ADC_INTEN_reg_t	INTEN	
ADC_FLAGS_reg_t	FLAGS	
ADC_TRM_reg_t	TRM	

7.28. Referencia del Archivo includes/hri/HRI_CTIMER.h

Definiciones a nivel de registros del periferico CTIMER (LPC845)

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

Dependencia gráfica adjunta para HRI_CTIMER.h:

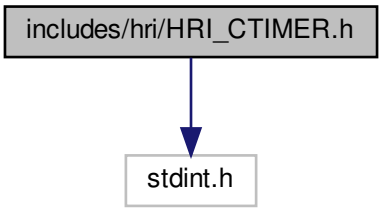
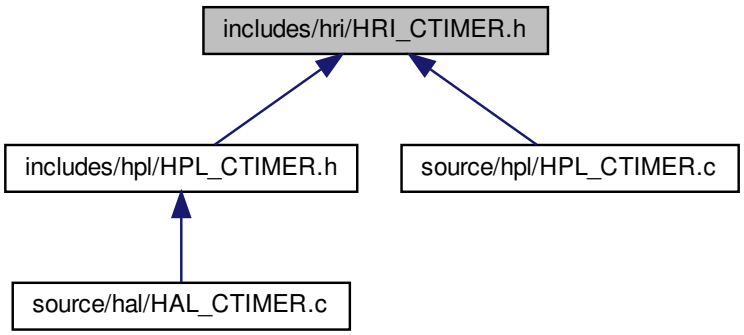


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct CTIMER_IR_reg_t
- struct CTIMER_TCR_reg_t
- struct CTIMER_TC_reg_t
- struct CTIMER_PR_reg_t
- struct CTIMER_PC_reg_t
- struct CTIMER_MCR_reg_t
- struct CTIMER_MR_reg_t
- struct CTIMER_CCR_reg_t
- struct CTIMER_CR_reg_t
- struct CTIMER_EMR_reg_t
- struct CTIMER_CTCR_reg_t
- struct CTIMER_PWMC_reg_t
- struct CTIMER_MSR_reg_t
- struct CTIMER_per_t

defines

- #define **CTIMER_BASE** 0x40038000

7.28.1. Descripción detallada

Definiciones a nivel de registros del periferico CTIMER (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.28.2. Documentación de las estructuras de datos

7.28.2.1. struct CTIMER_IR_reg_t

Campos de datos

uint32_t	MR0INT: 1	
uint32_t	MR1INT: 1	
uint32_t	MR2INT: 1	
uint32_t	MR3INT: 1	
uint32_t	CR0INT: 1	
uint32_t	CR1INT: 1	
uint32_t	CR2INT: 1	
uint32_t	CR3INT: 1	
uint32_t	__pad0__: 24	

7.28.2.2. struct CTIMER_TCR_reg_t

Campos de datos

uint32_t	CEN: 1	
uint32_t	CRST: 1	
uint32_t	__pad0__: 30	

7.28.2.3. struct CTIMER_TC_reg_t

Campos de datos

uint32_t	TCVAL	

7.28.2.4. struct CTIMER_PR_reg_t

Campos de datos

uint32_t	PRVAL	

7.28.2.5. struct CTIMER_PC_reg_t

Campos de datos

uint32_t	PCVAL	

7.28.2.6. struct CTIMER_MCR_reg_t

Campos de datos

uint32_t	MR0I: 1	
uint32_t	MR0R: 1	
uint32_t	MR0S: 1	
uint32_t	MR1I: 1	
uint32_t	MR1R: 1	
uint32_t	MR1S: 1	
uint32_t	MR2I: 1	
uint32_t	MR2R: 1	
uint32_t	MR2S: 1	
uint32_t	MR3I: 1	
uint32_t	MR3R: 1	
uint32_t	MR3S: 1	
uint32_t	__pad0__: 12	
uint32_t	MR0RL: 1	
uint32_t	MR1RL: 1	
uint32_t	MR2RL: 1	
uint32_t	MR3RL: 1	
uint32_t	__pad1__: 4	

7.28.2.7. struct CTIMER_MR_reg_t

Campos de datos

uint32_t	MATCH	

7.28.2.8. struct CTIMER_CCR_reg_t

Campos de datos

uint32_t	CAP0RE: 1	
uint32_t	CAP0FE: 1	
uint32_t	CAP0I: 1	
uint32_t	CAP1RE: 1	
uint32_t	CAP1FE: 1	
uint32_t	CAP1I: 1	
uint32_t	CAP2RE: 1	
uint32_t	CAP2FE: 1	
uint32_t	CAP2I: 1	
uint32_t	CAP3RE: 1	
uint32_t	CAP3FE: 1	
uint32_t	CAP3I: 1	
uint32_t	__pad0__: 20	

7.28.2.9. struct CTIMER_CR_reg_t

Campos de datos

uint32_t	CAP	

7.28.2.10. struct CTIMER_EMR_reg_t

Campos de datos

uint32_t	EM0: 1	
uint32_t	EM1: 1	
uint32_t	EM2: 1	
uint32_t	EM3: 1	
uint32_t	EMC0: 2	
uint32_t	EMC1: 2	
uint32_t	EMC2: 2	
uint32_t	EMC3: 2	
uint32_t	__pad0__: 20	

7.28.2.11. struct CTIMER_CTCR_reg_t

Campos de datos

uint32_t	CTMODE: 2	
uint32_t	CINSEL: 2	
uint32_t	ENCC: 1	
uint32_t	SELCC: 3	
uint32_t	__pad0__: 24	

7.28.2.12. struct CTIMER_PWMC_reg_t

Campos de datos

uint32_t	PWMEN0: 1	
uint32_t	PWMEN1: 1	
uint32_t	PWMEN2: 1	
uint32_t	PWMEN3: 1	
uint32_t	__pad0__: 28	

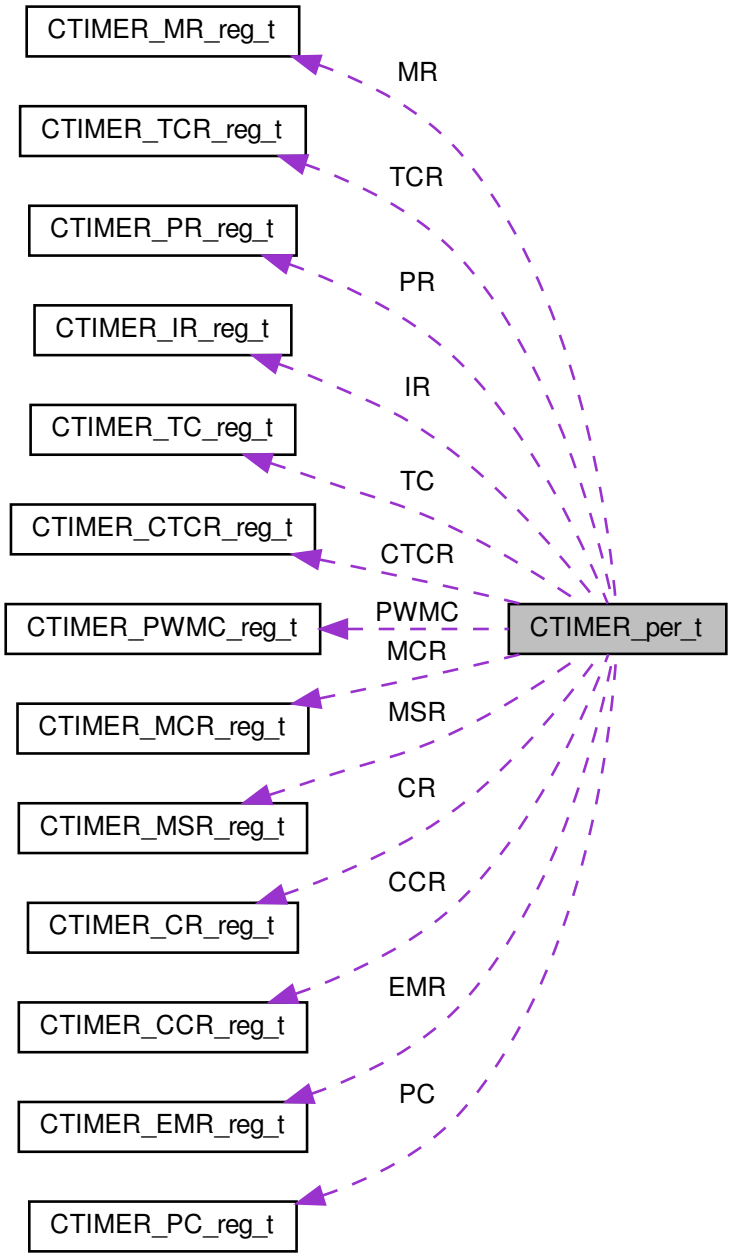
7.28.2.13. struct CTIMER_MSR_reg_t

Campos de datos

uint32_t	SHADOW	

7.28.2.14. struct CTIMER_per_t

Diagrama de colaboración para CTIMER_per_t:



Campos de datos

CTIMER_IR_reg_t	IR	
CTIMER_TCR_reg_t	TCR	
CTIMER_TC_reg_t	TC	

Campos de datos

CTIMER_PR_reg_t	PR	
CTIMER_PC_reg_t	PC	
CTIMER_MCR_reg_t	MCR	
CTIMER_MR_reg_t	MR[4]	
CTIMER_CCR_reg_t	CCR	
const CTIMER_CR_reg_t	CR[4]	
CTIMER_EMR_reg_t	EMR	
const uint32_t	RESERVED[12]	
CTIMER_CTCR_reg_t	CTCR	
CTIMER_PWMC_reg_t	PWMC	
CTIMER_MSR_reg_t	MSR[4]	

7.29. Referencia del Archivo includes/hri/HRI_DAC.h

Declaraciones a nivel de registros del DAC (LPC845)

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

Dependencia gráfica adjunta para HRI_DAC.h:

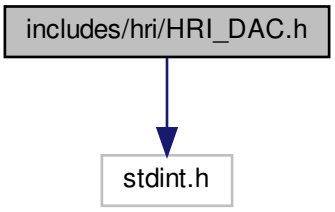
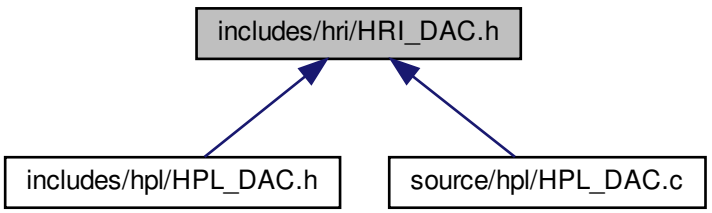


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct DAC_CR_reg_t
- struct DAC_CTRL_reg_t
- struct DAC_CNTVAL_reg_t
- struct DAC_per_t

defines

- #define **DAC0_BASE** 0x40014000
- #define **DAC1_BASE** 0x40018000

7.29.1. Descripción detallada

Declaraciones a nivel de registros del DAC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.29.2. Documentación de las estructuras de datos

7.29.2.1. struct DAC_CR_reg_t

Campos de datos

uint32_t	__pad0__: 6	
uint32_t	VALUE: 10	
uint32_t	BIAS: 1	
uint32_t	__pad1__: 16	

7.29.2.2. struct DAC_CTRL_reg_t

Campos de datos

uint32_t	INT_DMA_REQ: 1	
uint32_t	DBLBUF_ENA: 1	
uint32_t	CNT_ENA: 1	
uint32_t	DMA_ENA: 1	

Campos de datos

uint32_t	__pad0__: 28	

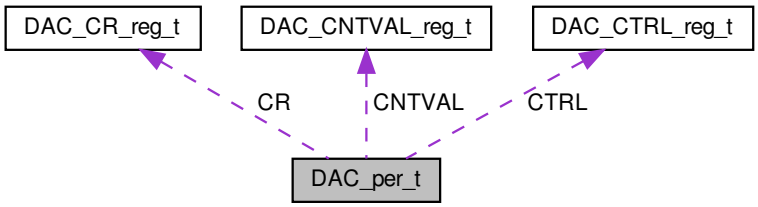
7.29.2.3. struct DAC_CNTVAL_reg_t

Campos de datos

uint32_t	VALUE: 16	
uint32_t	__pad0__: 16	

7.29.2.4. struct DAC_per_t

Diagrama de colaboración para DAC_per_t:



Campos de datos

DAC_CR_reg_t	CR	
DAC_CTRL_reg_t	CTRL	
DAC_CNTVAL_reg_t	CNTVAL	

7.30. Referencia del Archivo includes/hri/HRI_GPIO.h

Definiciones a nivel de registros del modulo GPIO (LPC845)

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

Dependencia gráfica adjunta para HRI_GPIO.h:

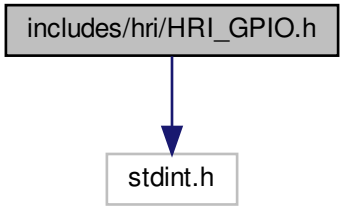
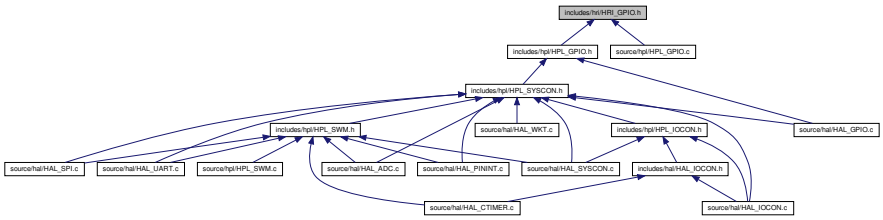


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct GPIO_B_reg_t
- struct GPIO_W_reg_t
- struct GPIO_DIR_reg_t
- struct GPIO_MASK_reg_t
- struct GPIO_PIN_reg_t
- struct GPIO_MPIN_reg_t
- struct GPIO_SET_reg_t
- struct GPIO_CLR_reg_t
- struct GPIO_NOT_reg_t
- struct GPIO_DIRSET_reg_t
- struct GPIO_DIRCLR_reg_t
- struct GPIO_DIRNOT_reg_t
- struct GPIO_per_t

defines

- #define **GPIO_BASE** 0xA0000000

7.30.1. Descripción detallada

Definiciones a nivel de registros del modulo GPIO (LPC845)

Definiciones a nivel de registros del modulo IOCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.30.2. Documentación de las estructuras de datos

7.30.2.1. struct GPIO_B_reg_t

Campos de datos

uint8_t	PBYTE: 1	
uint8_t	__pad0__: 7	

7.30.2.2. struct GPIO_W_reg_t

Campos de datos

uint32_t	PWORD	

7.30.2.3. struct GPIO_DIR_reg_t

Campos de datos

uint32_t	DIRP	

7.30.2.4. struct GPIO_MASK_reg_t

Campos de datos

uint32_t	MASKP	

7.30.2.5. struct GPIO_PIN_reg_t

Campos de datos

uint32_t	PORT	

7.30.2.6. struct GPIO_MPIN_reg_t

Campos de datos

uint32_t	MPORTP	

7.30.2.7. struct GPIO_SET_reg_t

Campos de datos

uint32_t	SETP	

7.30.2.8. struct GPIO_CLR_reg_t

Campos de datos

uint32_t	CLRP	

7.30.2.9. struct GPIO_NOT_reg_t

Campos de datos

uint32_t	NOTP	

7.30.2.10. struct GPIO_DIRSET_reg_t

Campos de datos

uint32_t	DIRSETP	

7.30.2.11. struct GPIO_DIRCLR_reg_t

Campos de datos

uint32_t	DIRCLRP	

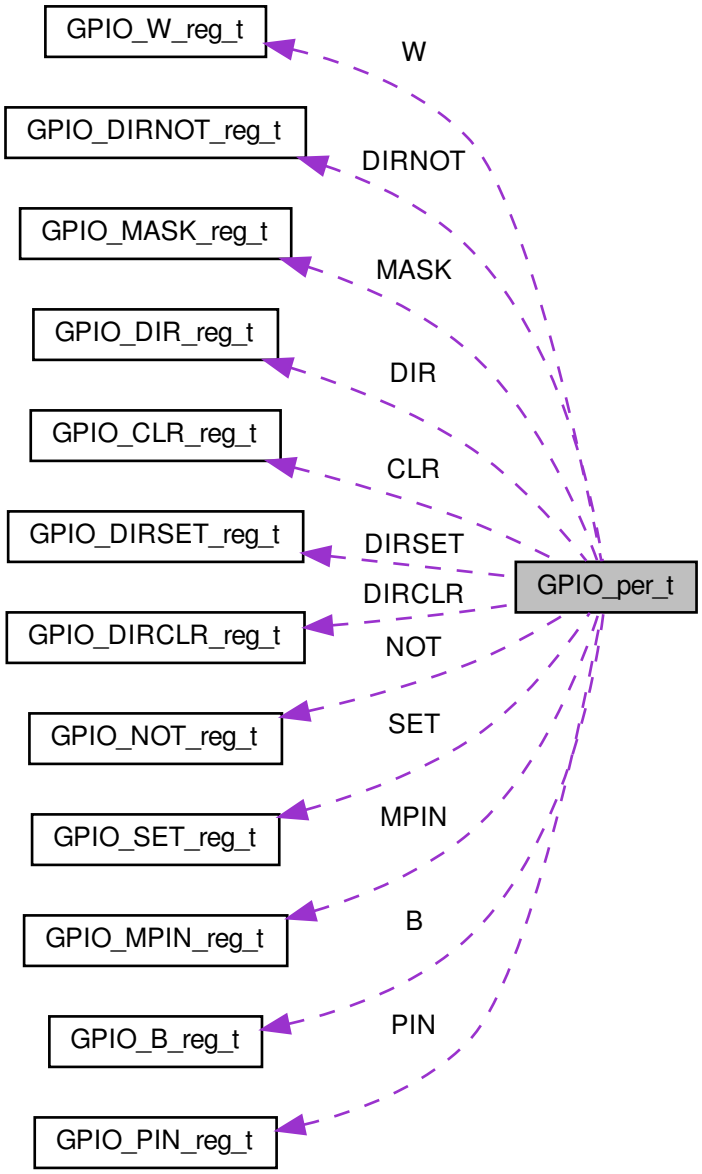
7.30.2.12. struct GPIO_DIRNOT_reg_t

Campos de datos

uint32_t	DIRNOTP	

7.30.2.13. struct GPIO_per_t

Diagrama de colaboración para GPIO_per_t:



Campos de datos

GPIO_B_reg_t	B[54]	

Campos de datos

const uint8_t	RESERVED_1[0xFCA]	
GPIO_W_reg_t	W[54]	
const uint8_t	RESERVED_2[0xF28]	
GPIO_DIR_reg_t	DIR[2]	
const uint8_t	RESERVED_3[0x78]	
GPIO_MASK_reg_t	MASK[2]	
const uint8_t	RESERVED_4[0x78]	
GPIO_PIN_reg_t	PIN[2]	
const uint8_t	RESERVED_5[0x78]	
GPIO_MPIN_reg_t	MPIN[2]	
const uint8_t	RESERVED_6[0x75]	
GPIO_SET_reg_t	SET[2]	
const uint8_t	RESERVED_7[0x78]	
GPIO_CLR_reg_t	CLR[2]	
const uint8_t	RESERVED_8[0x78]	
GPIO_NOT_reg_t	NOT[2]	
const uint8_t	RESERVED_9[0x78]	
GPIO_DIRSET_reg_t	DIRSET[2]	
const uint8_t	RESERVED_10[0x78]	
GPIO_DIRCLR_reg_t	DIRCLR[2]	
const uint8_t	RESERVED_11[0x78]	
GPIO_DIRNOT_reg_t	DIRNOT[2]	

7.31. Referencia del Archivo includes/hri/HRI_MRT.h

Definiciones a nivel de registros del periferico MRT (LPC845)

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

Dependencia gráfica adjunta para HRI_MRT.h:

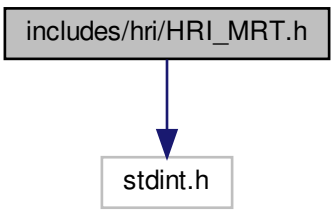
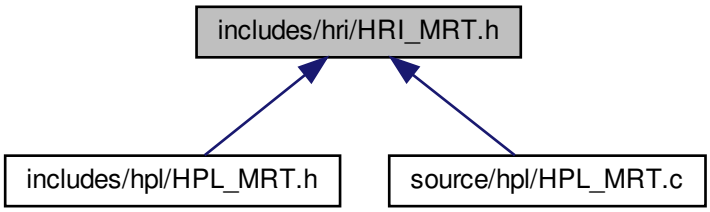


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct MRT_INTVAL_reg_t
- struct MRT_TIMER_reg_t
- struct MRT_CTRL_reg_t
- struct MRT_STAT_reg_t
- struct MRT_IDLE_CH_reg_t
- struct MRT_IRQ_FLAG_reg_t
- struct MRT_CHN_reg_t
- struct MRT_per_t

defines

- #define **MRT_BASE** 0x40004000

7.31.1. Descripción detallada

Definiciones a nivel de registros del periferico MRT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.31.2. Documentación de las estructuras de datos

7.31.2.1. struct MRT_INTVAL_reg_t

Campos de datos

uint32_t	LVALUE: 31	
uint32_t	LOAD: 1	

7.31.2.2. struct MRT_TIMER_reg_t

Campos de datos

uint32_t	VALUE: 31	
uint32_t	__pad0__: 1	

7.31.2.3. struct MRT_CTRL_reg_t

Campos de datos

uint32_t	INTEN: 1	
uint32_t	MODE: 2	
uint32_t	__pad0__: 29	

7.31.2.4. struct MRT_STAT_reg_t

Campos de datos

uint32_t	INTFLAG: 1	
uint32_t	RUN: 1	
uint32_t	__pad0__: 30	

7.31.2.5. struct MRT_IDLE_CH_reg_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	CHAN: 4	
uint32_t	__pad1__: 24	

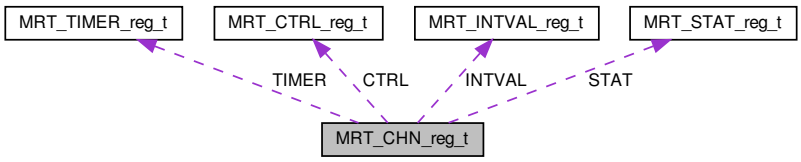
7.31.2.6. struct MRT_IRQ_FLAG_reg_t

Campos de datos

uint32_t	CFLAG0: 1	
uint32_t	CFLAG1: 1	
uint32_t	CFLAG2: 1	
uint32_t	CFLAG3: 1	
uint32_t	__pad0__: 28	

7.31.2.7. struct MRT_CHN_reg_t

Diagrama de colaboración para MRT_CHN_reg_t:

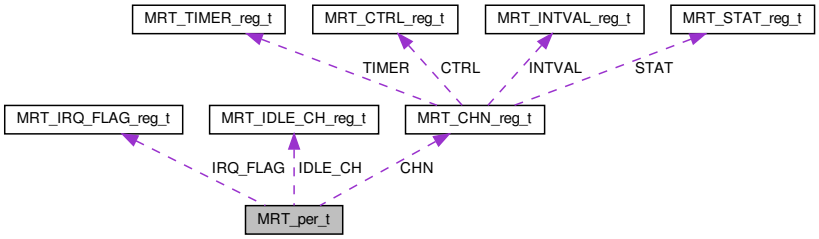


Campos de datos

MRT_INTVAL_reg_t	INTVAL	
const MRT_TIMER_reg_t	TIMER	
MRT_CTRL_reg_t	CTRL	
MRT_STAT_reg_t	STAT	

7.31.2.8. struct MRT_per_t

Diagrama de colaboración para MRT_per_t:



Campos de datos

MRT_CHN_reg_t	CHN[4]	
const uint32_t	RESERVED	
const MRT_IDLE_CH_reg_t	IDLE_CH	
MRT_IRQ_FLAG_reg_t	IRQ_FLAG	

7.32. Referencia del Archivo includes/hri/HRI_NVIC.h

Definiciones a nivel de registros del modulo NVIC (LPC845)

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

Dependencia gráfica adjunta para HRI_NVIC.h:

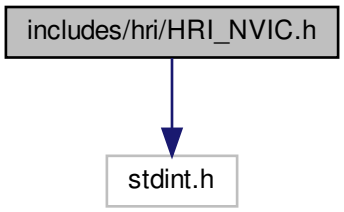
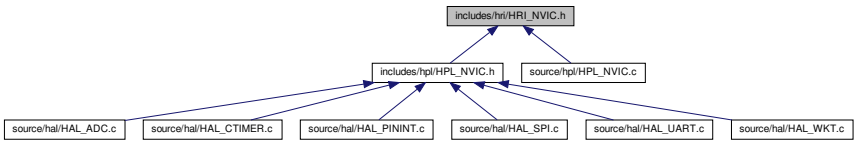


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct NVIC_ISER0_reg_t
- struct NVIC_ICER0_reg_t
- struct NVIC_ISPR0_reg_t
- struct NVIC_ICPR0_reg_t
- struct NVIC_IABR0_reg_t
- struct NVIC_IPR0_reg_t
- struct NVIC_IPR1_reg_t
- struct NVIC_IPR2_reg_t
- struct NVIC_IPR3_reg_t
- struct NVIC_IPR4_reg_t
- struct NVIC_IPR5_reg_t
- struct NVIC_IPR6_reg_t
- struct NVIC_IPR7_reg_t
- struct NVIC_per_t

defines

- #define **NVIC_BASE** 0xE000E000

7.32.1. Descripción detallada

Definiciones a nivel de registros del modulo NVIC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.32.2. Documentación de las estructuras de datos

7.32.2.1. struct NVIC_ISER0_reg_t

Campos de datos

uint32_t	ISE_SPI0: 1	
uint32_t	ISE_SPI1: 1	
uint32_t	ISE_DAC0: 1	
uint32_t	ISE_UART0: 1	
uint32_t	ISE_UART1: 1	
uint32_t	ISE_UART2: 1	
uint32_t	__pad0__: 1	
uint32_t	ISE_I2C1: 1	
uint32_t	ISE_I2C0: 1	
uint32_t	ISE_SCT: 1	
uint32_t	ISE_MRT: 1	
uint32_t	ISE_CMP: 1	
uint32_t	ISE_WDT: 1	
uint32_t	ISE_BOD: 1	
uint32_t	ISE_FLASH: 1	
uint32_t	ISE_WKT: 1	
uint32_t	ISE_ADC_SEQA: 1	
uint32_t	ISE_ADC_SEQB: 1	
uint32_t	ISE_ADC_THCMP: 1	
uint32_t	ISE_ADC_OVR: 1	
uint32_t	ISE_SDMA: 1	
uint32_t	ISE_I2C2: 1	
uint32_t	ISE_I2C3: 1	
uint32_t	ISE_CT32B0: 1	
uint32_t	ISE_PININT0: 1	
uint32_t	ISE_PININT1: 1	
uint32_t	ISE_PININT2: 1	
uint32_t	ISE_PININT3: 1	
uint32_t	ISE_PININT4: 1	
uint32_t	ISE_PININT5: 1	
uint32_t	ISE_PININT6: 1	
uint32_t	ISE_PININT7: 1	

7.32.2.2. struct NVIC_ICER0_reg_t

Campos de datos

uint32_t	ICE_SPI0: 1	
uint32_t	ICE_SPI1: 1	
uint32_t	ICE_DAC0: 1	
uint32_t	ICE_UART0: 1	
uint32_t	ICE_UART1: 1	
uint32_t	ICE_UART2: 1	
uint32_t	__pad0__: 1	
uint32_t	ICE_I2C1: 1	
uint32_t	ICE_I2C0: 1	
uint32_t	ICE_SCT: 1	
uint32_t	ICE_MRT: 1	
uint32_t	ICE_CMP: 1	
uint32_t	ICE_WDT: 1	
uint32_t	ICE_BOD: 1	
uint32_t	ICE_FLASH: 1	
uint32_t	ICE_WKT: 1	
uint32_t	ICE_ADC_SEQA: 1	
uint32_t	ICE_ADC_SEQB: 1	
uint32_t	ICE_ADC_THCMP: 1	
uint32_t	ICE_ADC_OVR: 1	
uint32_t	ICE_SDMA: 1	
uint32_t	ICE_I2C2: 1	
uint32_t	ICE_I2C3: 1	
uint32_t	ICE_CT32b0: 1	
uint32_t	ICE_PININT0: 1	
uint32_t	ICE_PININT1: 1	
uint32_t	ICE_PININT2: 1	
uint32_t	ICE_PININT3: 1	
uint32_t	ICE_PININT4: 1	
uint32_t	ICE_PININT5: 1	
uint32_t	ICE_PININT6: 1	
uint32_t	ICE_PININT7: 1	

7.32.2.3. struct NVIC_ISPR0_reg_t

Campos de datos

uint32_t	ISP_SPI0: 1	
uint32_t	ISP_SPI1: 1	
uint32_t	ISP_DAC0: 1	
uint32_t	ISP_UART0: 1	
uint32_t	ISP_UART1: 1	
uint32_t	ISP_UART2: 1	
uint32_t	__pad0__: 1	
uint32_t	ISP_I2C1: 1	
uint32_t	ISP_I2C0: 1	

Campos de datos

uint32_t	ISP_SCT: 1	
uint32_t	ISP_MRT: 1	
uint32_t	ISP_CMP: 1	
uint32_t	ISP_WDT: 1	
uint32_t	ISP_BOD: 1	
uint32_t	ISP_FLASH: 1	
uint32_t	ISP_WKT: 1	
uint32_t	ISP_ADC_SEQA: 1	
uint32_t	ISP_ADC_SEQB: 1	
uint32_t	ISP_ADC_THCMP: 1	
uint32_t	ISP_ADC_OVR: 1	
uint32_t	ISP_SDMA: 1	
uint32_t	ISP_I2C2: 1	
uint32_t	ISP_I2C3: 1	
uint32_t	ISP_CT32b0: 1	
uint32_t	ISP_PININT0: 1	
uint32_t	ISP_PININT1: 1	
uint32_t	ISP_PININT2: 1	
uint32_t	ISP_PININT3: 1	
uint32_t	ISP_PININT4: 1	
uint32_t	ISP_PININT5: 1	
uint32_t	ISP_PININT6: 1	
uint32_t	ISP_PININT7: 1	

7.32.2.4. struct NVIC_ICPR0_reg_t

Campos de datos

uint32_t	ICP_SPI0: 1	
uint32_t	ICP_SPI1: 1	
uint32_t	ICP_DAC0: 1	
uint32_t	ICP_UART0: 1	
uint32_t	ICP_UART1: 1	
uint32_t	ICP_UART2: 1	
uint32_t	__pad0__: 1	
uint32_t	ICP_I2C1: 1	
uint32_t	ICP_I2C0: 1	
uint32_t	ICP_SCT: 1	
uint32_t	ICP_MRT: 1	
uint32_t	ICP_CMP: 1	
uint32_t	ICP_WDT: 1	
uint32_t	ICP_BOD: 1	
uint32_t	ICP_FLASH: 1	
uint32_t	ICP_WKT: 1	
uint32_t	ICP_ADC_SEQA: 1	
uint32_t	ICP_ADC_SEQB: 1	
uint32_t	ICP_ADC_THCMP: 1	
uint32_t	ICP_ADC_OVR: 1	

Campos de datos

uint32_t	ICP_SDMA: 1	
uint32_t	ICP_I2C2: 1	
uint32_t	ICP_I2C3: 1	
uint32_t	ICP_CT32b0: 1	
uint32_t	ICP_PININT0: 1	
uint32_t	ICP_PININT1: 1	
uint32_t	ICP_PININT2: 1	
uint32_t	ICP_PININT3: 1	
uint32_t	ICP_PININT4: 1	
uint32_t	ICP_PININT5: 1	
uint32_t	ICP_PININT6: 1	
uint32_t	ICP_PININT7: 1	

7.32.2.5. struct NVIC_IABR0_reg_t

Campos de datos

uint32_t	IAB_SPI0: 1	
uint32_t	IAB_SPI1: 1	
uint32_t	IAB_DAC0: 1	
uint32_t	IAB_UART0: 1	
uint32_t	IAB_UART1: 1	
uint32_t	IAB_UART2: 1	
uint32_t	__pad0__: 1	
uint32_t	IAB_I2C1: 1	
uint32_t	IAB_I2C0: 1	
uint32_t	IAB_SCT: 1	
uint32_t	IAB_MRT: 1	
uint32_t	IAB_CMP: 1	
uint32_t	IAB_WDT: 1	
uint32_t	IAB_BOD: 1	
uint32_t	IAB_FLASH: 1	
uint32_t	IAB_WKT: 1	
uint32_t	IAB_ADC_SEQA: 1	
uint32_t	IAB_ADC_SEQB: 1	
uint32_t	IAB_ADC_THCMP: 1	
uint32_t	IAB_ADC_OVR: 1	
uint32_t	IAB_SDMA: 1	
uint32_t	IAB_I2C2: 1	
uint32_t	IAB_I2C3: 1	
uint32_t	IAB_CT32b0: 1	
uint32_t	IAB_PININT0: 1	
uint32_t	IAB_PININT1: 1	
uint32_t	IAB_PININT2: 1	
uint32_t	IAB_PININT3: 1	
uint32_t	IAB_PININT4: 1	
uint32_t	IAB_PININT5: 1	
uint32_t	IAB_PININT6: 1	
uint32_t	IAB_PININT7: 1	

7.32.2.6. struct NVIC_IPR0_reg_t

Campos de datos

uint32_t	__pad0__: 6	
uint32_t	IP_SPI0: 2	
uint32_t	__pad1__: 6	
uint32_t	IP_SPI1: 2	
uint32_t	__pad2__: 6	
uint32_t	IP_DAC0: 2	
uint32_t	__pad3__: 6	
uint32_t	IP_UART0: 2	

7.32.2.7. struct NVIC_IPR1_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_UART1: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_UART2: 2	
uint32_t	RESERVED_3: 6	
uint32_t	RESERVED_4: 8	
uint32_t	IP_I2C1: 2	

7.32.2.8. struct NVIC_IPR2_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_I2C0: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_SCT: 2	
uint32_t	RESERVED_3: 6	
uint32_t	IP_MRT: 2	
uint32_t	RESERVED_4: 6	
uint32_t	IP_CMP: 2	

7.32.2.9. struct NVIC_IPR3_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_WDT: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_BOD: 2	
uint32_t	RESERVED_3: 6	
uint32_t	IP_FLASH: 2	

Campos de datos

uint32_t	RESERVED_4: 6	
uint32_t	IP_WKT: 2	

7.32.2.10. struct NVIC_IPR4_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_ADC_SEQA: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_ADC_SEQB: 2	
uint32_t	RESERVED_3: 6	
uint32_t	IP_ADC_THCMP: 2	
uint32_t	RESERVED_4: 6	
uint32_t	ID_ADC_OVR: 2	

7.32.2.11. struct NVIC_IPR5_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_DMA: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_I2C2: 2	
uint32_t	RESERVED_3: 6	
uint32_t	IP_I2C3: 2	
uint32_t	RESERVED_4: 6	
uint32_t	IP_CT32B0: 2	

7.32.2.12. struct NVIC_IPR6_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_PININT0: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_PININT1: 2	
uint32_t	RESERVED_3: 6	
uint32_t	IP_PININT2: 2	
uint32_t	RESERVED_4: 6	
uint32_t	IP_PININT3: 2	

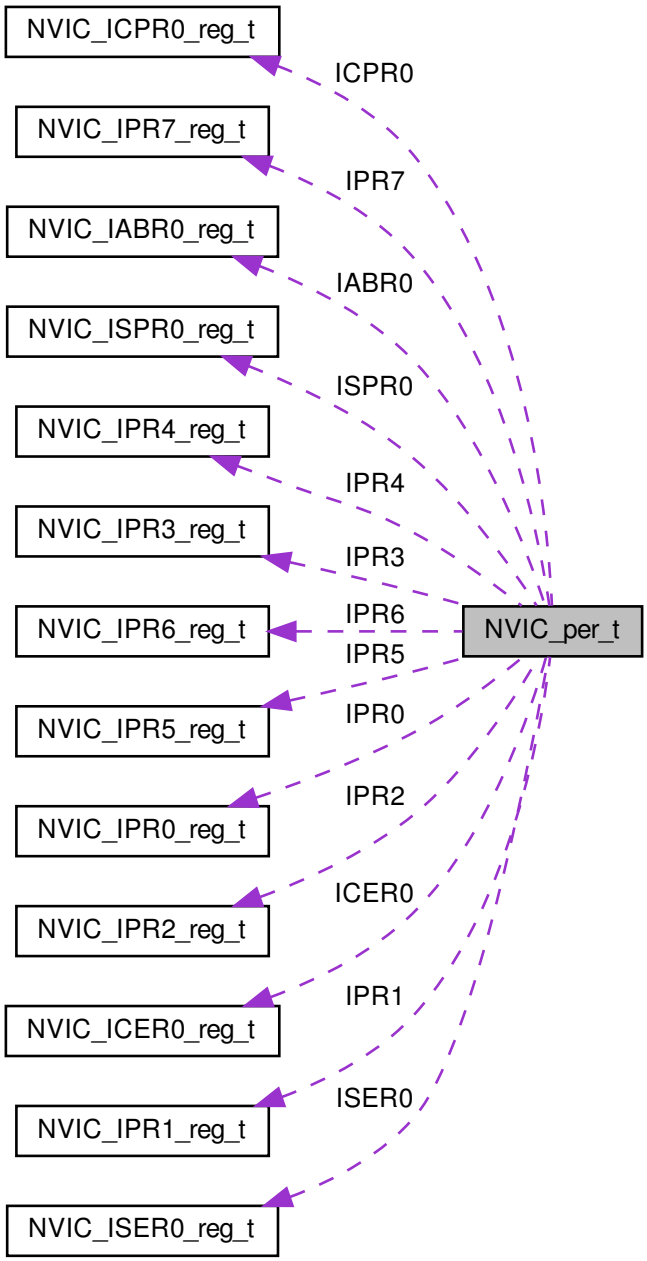
7.32.2.13. struct NVIC_IPR7_reg_t

Campos de datos

uint32_t	RESERVED_1: 6	
uint32_t	IP_PININT4: 2	
uint32_t	RESERVED_2: 6	
uint32_t	IP_PININT5: 2	
uint32_t	RESERVED_3: 6	
uint32_t	IP_PININT6: 2	
uint32_t	RESERVED_4: 6	
uint32_t	IP_PININT7: 2	

7.32.2.14. struct NVIC_per_t

Diagrama de colaboración para NVIC_per_t:



Campos de datos

const uint8_t	RESERVED_1[0x100]	
NVIC_ISER0_reg_t	ISER0	
const uint8_t	RESERVED_2[0x7C]	

Campos de datos

NVIC_ICER0_reg_t	ICER0	
const uint8_t	RESERVED_3[0x7C]	
NVIC_ISPR0_reg_t	ISPR0	
const uint8_t	RESERVED_4[0x7C]	
NVIC_ICPR0_reg_t	ICPR0	
const uint8_t	RESERVED_5[0x7C]	
const NVIC_IABR0_reg_t	IABR0	
const uint8_t	RESERVED_6[0xFC]	
NVIC_IPR0_reg_t	IPR0	
NVIC_IPR1_reg_t	IPR1	
NVIC_IPR2_reg_t	IPR2	
NVIC_IPR3_reg_t	IPR3	
NVIC_IPR4_reg_t	IPR4	
NVIC_IPR5_reg_t	IPR5	
NVIC_IPR6_reg_t	IPR6	
NVIC_IPR7_reg_t	IPR7	

7.33. Referencia del Archivo includes/hri/HRI_PININT.h

Definiciones a nivel de registros del modulo PININT (LPC845)

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

Dependencia gráfica adjunta para HRI_PININT.h:

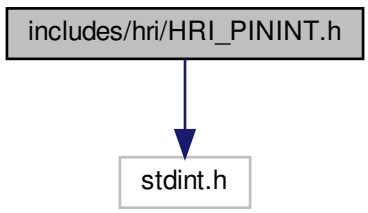
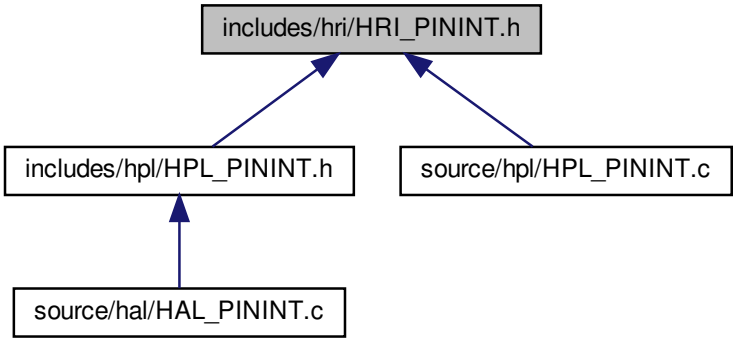


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct PININT_ISEL_reg_t
- struct PININT_IENR_reg_t
- struct PININT_SIENR_reg_t
- struct PININT_CIENR_reg_t
- struct PININT_IENF_reg_t
- struct PININT_SIENF_reg_t
- struct PININT_CIENF_reg_t
- struct PININT_RISE_reg_t
- struct PININT_FALL_reg_t
- struct PININT_IST_reg_t
- struct PININT_PMCTRL_reg_t
- struct PININT_PMSRC_reg_t
- struct PININT_PMCFG_reg_t
- struct PININT_per_t

defines

- #define PININT_BASE 0xA0004000

  Direccion base del PININT.

7.33.1. Descripción detallada

Definiciones a nivel de registros del modulo PININT (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.33.2. Documentación de las estructuras de datos

7.33.2.1. struct PININT_ISEL_reg_t

Campos de datos

uint32_t	PMODE: 8	
uint32_t	__pad0__: 24	

7.33.2.2. struct PININT_IENR_reg_t

Campos de datos

uint32_t	ENRL: 8	
uint32_t	__pad0__: 24	

7.33.2.3. struct PININT_SIENR_reg_t

Campos de datos

uint32_t	SETENRL: 8	
uint32_t	__pad0__: 24	

7.33.2.4. struct PININT_CIENR_reg_t

Campos de datos

uint32_t	CENRL: 8	
uint32_t	__pad0__: 24	

7.33.2.5. struct PININT_IENF_reg_t

Campos de datos

uint32_t	ENAF: 8	
uint32_t	__pad0__: 24	

7.33.2.6. struct PININT_SIENF_reg_t

Campos de datos

uint32_t	SETENAF: 8	
uint32_t	__pad0__: 24	

7.33.2.7. struct PININT_CIENF_reg_t

Campos de datos

uint32_t	CENAF: 8	
uint32_t	__pad0__: 24	

7.33.2.8. struct PININT_RISE_reg_t

Campos de datos

uint32_t	RDET: 8	
uint32_t	__pad0__: 24	

7.33.2.9. struct PININT_FALL_reg_t

Campos de datos

uint32_t	FDET: 8	
uint32_t	__pad0__: 24	

7.33.2.10. struct PININT_IST_reg_t

Campos de datos

uint32_t	PSTAT: 8	
uint32_t	__pad0__: 24	

7.33.2.11. struct PININT_PMCTRL_reg_t

Campos de datos

uint32_t	SEL_PMATCH: 1	
uint32_t	ENA_RXEV: 1	
uint32_t	__pad0__: 22	
uint32_t	PMAT: 8	

7.33.2.12. struct PININT_PMSRC_reg_t

Campos de datos

uint32_t	__pad0__: 8	
uint32_t	SRC0: 3	
uint32_t	SRC1: 3	
uint32_t	SRC2: 3	
uint32_t	SRC3: 3	

Campos de datos

uint32_t	SRC4: 3	
uint32_t	SRC5: 3	
uint32_t	SRC6: 3	
uint32_t	SRC7: 3	

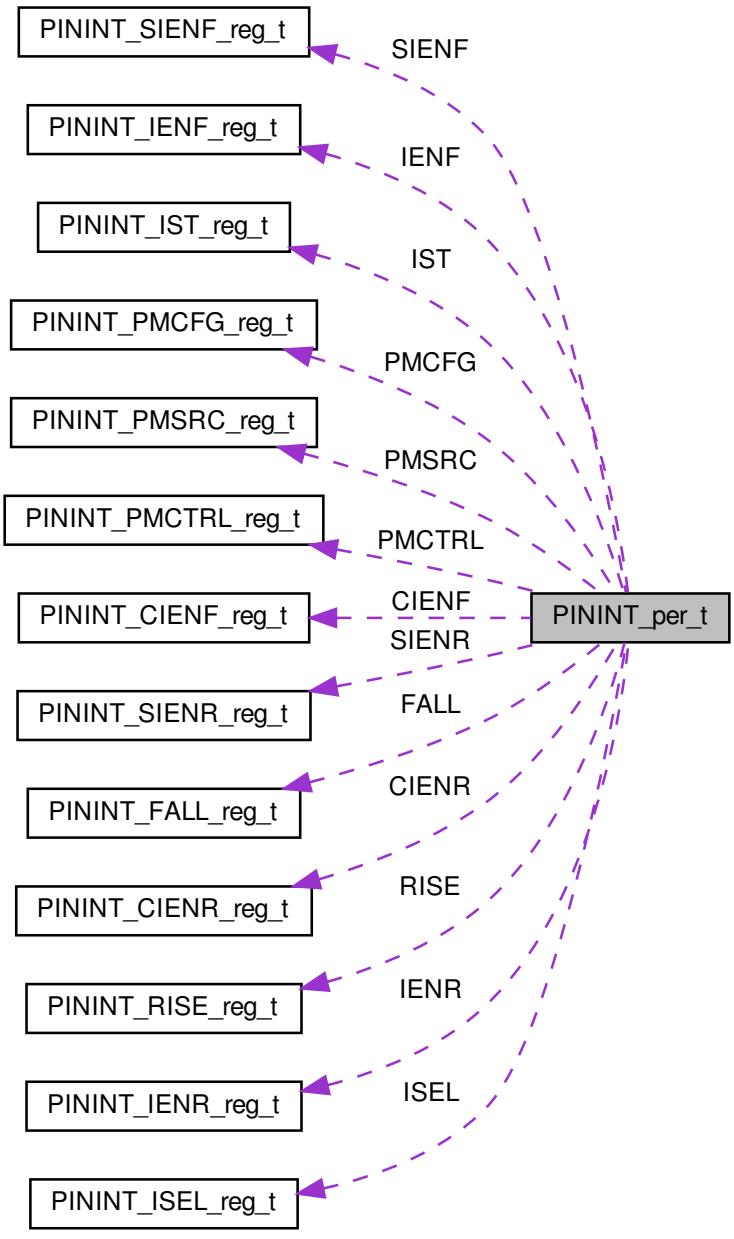
7.33.2.13. struct PININT_PMCFG_reg_t

Campos de datos

uint32_t	PROD_ENDPTS0: 1	
uint32_t	PROD_ENDPTS1: 1	
uint32_t	PROD_ENDPTS2: 1	
uint32_t	PROD_ENDPTS3: 1	
uint32_t	PROD_ENDPTS4: 1	
uint32_t	PROD_ENDPTS5: 1	
uint32_t	PROD_ENDPTS6: 1	
uint32_t	__pad0__: 1	
uint32_t	CFG0: 3	
uint32_t	CFG1: 3	
uint32_t	CFG2: 3	
uint32_t	CFG3: 3	
uint32_t	CFG4: 3	
uint32_t	CFG5: 3	
uint32_t	CFG6: 3	
uint32_t	CFG7: 3	

7.33.2.14. struct PININT_per_t

Diagrama de colaboración para PININT_per_t:



Campos de datos

PININT_ISEL_reg_t	ISEL	
PININT_IENR_reg_t	IENR	
PININT_SIENR_reg_t	SIENR	
PININT_CIENR_reg_t	CIENR	

Campos de datos

PININT_IENF_reg_t	IENF	
PININT_SIENF_reg_t	SIENF	
PININT_CIENF_reg_t	CIENF	
PININT_RISE_reg_t	RISE	
PININT_FALL_reg_t	FALL	
PININT_IST_reg_t	IST	
PININT_PMCTRL_reg_t	PMCTRL	
PININT_PMSRC_reg_t	PMSRC	
PININT_PMCFG_reg_t	PMCFG	

7.34. Referencia del Archivo includes/hri/HRI_PMU.h

Definiciones a nivel de registros del modulo PMU (LPC845)

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

Dependencia gráfica adjunta para HRI_PMU.h:

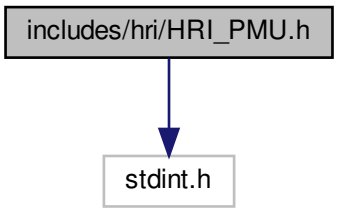
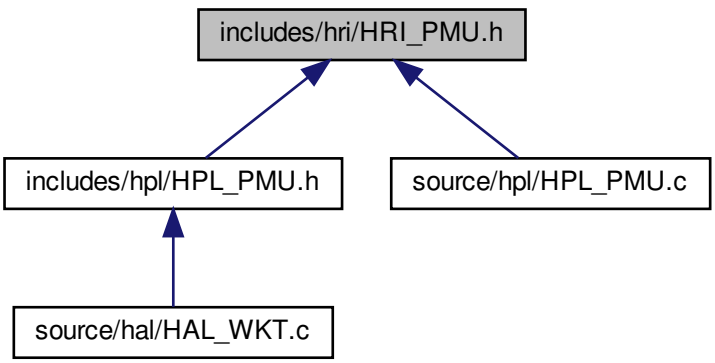


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct SCR_reg_t
- struct PMU_PCON_reg_t
- struct PMU_GPREG_reg_t
- struct PMU_DPDCTRL_reg_t
- struct PMU_per_t

defines

- #define **PMU_BASE** 0x40020000
- #define **SCR_REG_BASE** 0xE000ED10

7.34.1. Descripción detallada

Definiciones a nivel de registros del modulo PMU (LPC845)

Autor

Augusto Santini

Fecha

4/2020

Versión

1.0

7.34.2. Documentación de las estructuras de datos

7.34.2.1. struct SCR_reg_t

Campos de datos

uint32_t	__pad0__: 1	
uint32_t	SLEEPONEXIT: 1	
uint32_t	SLEEPDEEP: 1	
uint32_t	__pad1__: 1	
uint32_t	SEVONPEND: 1	
uint32_t	__pad2__: 27	

7.34.2.2. struct PMU_PCON_reg_t

Campos de datos

uint32_t	PM: 2	

Campos de datos

uint32_t	NODPD: 1	
uint32_t	__pad0__: 4	
uint32_t	SLEEPFLAG: 1	
uint32_t	__pad1__: 2	
uint32_t	DPDFLAG: 1	
uint32_t	__pad2__: 21	

7.34.2.3. struct PMU_GPREG_reg_t

Campos de datos

uint32_t	GPDATA	

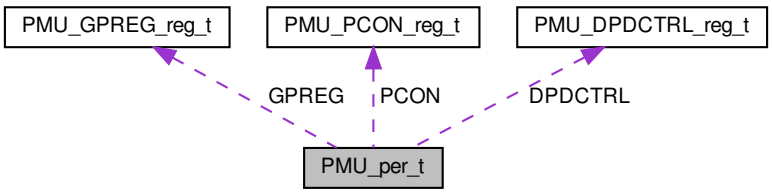
7.34.2.4. struct PMU_DPDCTRL_reg_t

Campos de datos

uint32_t	WAKEUPHYS: 1	
uint32_t	WAKEPAD_DISABLE: 1	
uint32_t	LPOSCEN: 1	
uint32_t	LPOSCDPDEN: 1	
uint32_t	WAKEUPCLKHYS: 1	
uint32_t	WAKECLKPAD_DISABLE: 1	
uint32_t	RESETHYS: 1	
uint32_t	RESET_DISABLE: 1	
uint32_t	__pad0__: 24	

7.34.2.5. struct PMU_per_t

Diagrama de colaboración para PMU_per_t:



Campos de datos

PMU_PCON_reg_t	PCON	
PMU_GPREG_reg_t	GPREG[4]	
PMU_DPDCTRL_reg_t	DPDCTRL	

7.35. Referencia del Archivo includes/hri/HRI_SPI.h

Definiciones a nivel de registros del periferico SPI (LPC845)

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

Dependencia gráfica adjunta para HRI_SPI.h:

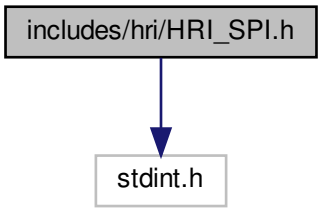
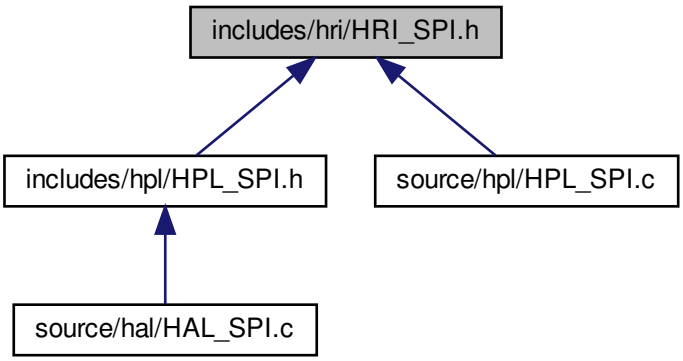


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct SPI_CFG_reg_t
- struct SPI_DLY_reg_t
- struct SPI_STAT_reg_t
- struct SPI_INTENSET_reg_t
- struct SPI_INTENCLR_reg_t
- struct SPI_RXDAT_reg_t
- struct SPI_TXDATCTL_reg_t
- struct SPI_TXDAT_reg_t
- struct SPI_TXCTL_reg_t
- struct SPI_DIV_reg_t
- struct SPI_INTSTAT_reg_t
- struct SPI_per_t

defines

- #define **SPI0_BASE** 0x40058000
- #define **SPI1_BASE** 0x4005C000

7.35.1. Descripción detallada

Definiciones a nivel de registros del periferico SPI (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.35.2. Documentación de las estructuras de datos

7.35.2.1. struct SPI_CFG_reg_t

Campos de datos

uint32_t	ENABLE: 1	
uint32_t	__pad0__: 1	
uint32_t	MASTER: 1	
uint32_t	LSBF: 1	
uint32_t	CPHA: 1	
uint32_t	CPOL: 1	
uint32_t	__pad1__: 1	
uint32_t	LOOP: 1	
uint32_t	SPOL0: 1	
uint32_t	SPOL1: 1	
uint32_t	SPOL2: 1	
uint32_t	SPOL3: 1	
uint32_t	__pad2__: 20	

7.35.2.2. struct SPI_DLY_reg_t

Campos de datos

uint32_t	PRE_DELAY: 4	
uint32_t	POST_DELAY: 4	

Campos de datos

uint32_t	FRAME_DELAY: 4	
uint32_t	TRANSFER_DELAY: 4	
uint32_t	__pad0__: 16	

7.35.2.3. struct SPI_STAT_reg_t

Campos de datos

const uint32_t	RXRDY: 1	
const uint32_t	TXRDY: 1	
uint32_t	RXOV: 1	
uint32_t	TXUR: 1	
uint32_t	SSA: 1	
uint32_t	SSD: 1	
const uint32_t	STALLED: 1	
uint32_t	ENDTRANSFER: 1	
const uint32_t	MSTIDLE: 1	
uint32_t	__pad0__: 21	

7.35.2.4. struct SPI_INTENSET_reg_t

Campos de datos

uint32_t	RXRDYEN: 1	
uint32_t	TXRDYEN: 1	
uint32_t	RXOVEN: 1	
uint32_t	TXUREN: 1	
uint32_t	SSAEN: 1	
uint32_t	SSDEN: 1	
uint32_t	__pad0__: 26	

7.35.2.5. struct SPI_INTENCLR_reg_t

Campos de datos

uint32_t	RXRDYEN: 1	
uint32_t	TXRDYEN: 1	
uint32_t	RXOVEN: 1	
uint32_t	TXUREN: 1	
uint32_t	SSAEN: 1	
uint32_t	SSDEN: 1	
uint32_t	__pad0__: 26	

7.35.2.6. struct SPI_RXDAT_reg_t

Campos de datos

uint32_t	RXDAT: 16	
uint32_t	RXSSEL0_N: 1	
uint32_t	RXSSEL1_N: 1	
uint32_t	RXSSEL2_N: 1	
uint32_t	RXSSEL3_N: 1	
uint32_t	SOT: 1	
uint32_t	__pad0__: 11	

7.35.2.7. struct SPI_TXDATCTL_reg_t

Campos de datos

uint32_t	TXDAT: 16	
uint32_t	TXSSEL0_N: 1	
uint32_t	TXSSEL1_N: 1	
uint32_t	TXSSEL2_N: 1	
uint32_t	TXSSEL3_N: 1	
uint32_t	EOT: 1	
uint32_t	EOf: 1	
uint32_t	RXIGNORE: 1	
uint32_t	__pad0__: 1	
uint32_t	LEN: 4	
uint32_t	__pad1__: 4	

7.35.2.8. struct SPI_TXDAT_reg_t

Campos de datos

uint32_t	DATA: 16	
uint32_t	__pad0__: 16	

7.35.2.9. struct SPI_TXCTL_reg_t

Campos de datos

uint32_t	__pad0__: 16	
uint32_t	TXSSEL0_N: 1	
uint32_t	TXSSEL1_N: 1	
uint32_t	TXSSEL2_N: 1	
uint32_t	TXSSEL3_N: 1	
uint32_t	EOT: 1	
uint32_t	EOf: 1	
uint32_t	RXIGNORE: 1	
uint32_t	__pad1__: 1	
uint32_t	LEN: 4	
uint32_t	__pad2__: 4	

7.35.2.10. struct SPI_DIV_reg_t

Campos de datos

uint32_t	DIVVAL: 16	
uint32_t	__pad0__: 16	

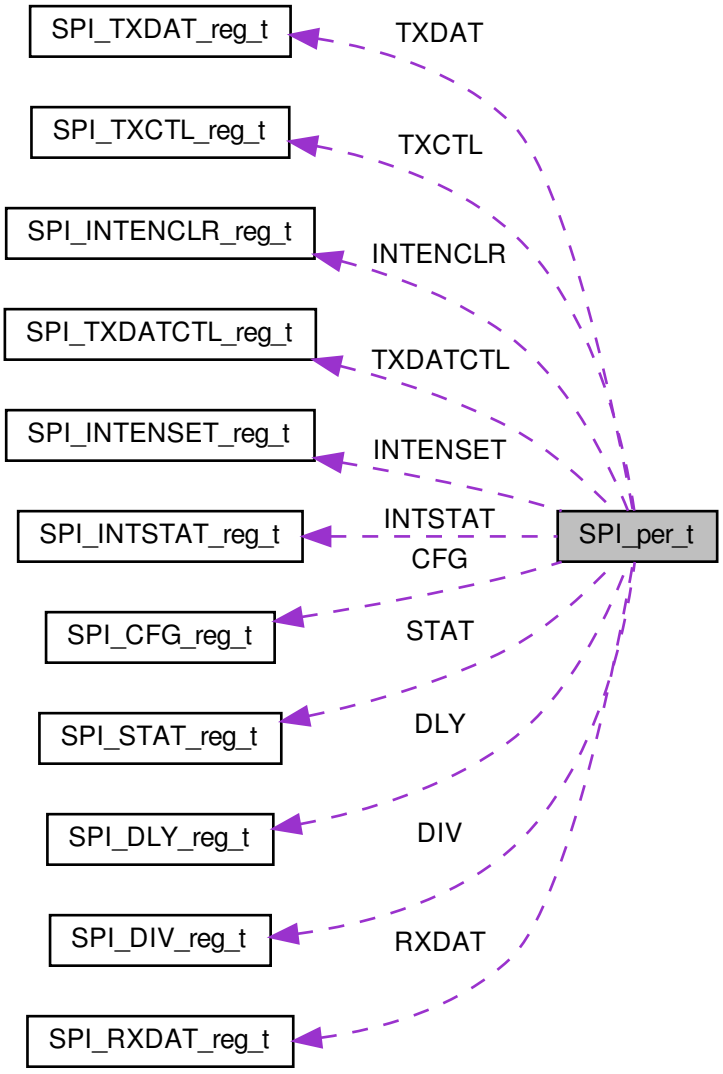
7.35.2.11. struct SPI_INTSTAT_reg_t

Campos de datos

uint32_t	RXRDY: 1	
uint32_t	TXRDY: 1	
uint32_t	RXOV: 1	
uint32_t	TXUR: 1	
uint32_t	SSA: 1	
uint32_t	SSD: 1	
uint32_t	__pad0__: 26	

7.35.2.12. struct SPI_per_t

Diagrama de colaboración para SPI_per_t:



Campos de datos

SPI_CFG_reg_t	CFG	
SPI_DLY_reg_t	DLY	
SPI_STAT_reg_t	STAT	
SPI_INTENSET_reg_t	INTENSET	
SPI_INTENCLR_reg_t	INTENCLR	
const SPI_RXDAT_reg_t	RXDAT	
SPI_TXDATCTL_reg_t	TXDATCTL	
SPI_TXDAT_reg_t	TXDAT	

Campos de datos

SPI_TXCTL_reg_t	TXCTL	
SPI_DIV_reg_t	DIV	
const SPI_INTSTAT_reg_t	INTSTAT	

7.36. Referencia del Archivo includes/hri/HRI_SWM.h

Definiciones a nivel de registros del modulo SWM (LPC845)

`#include <stdint.h>`
Dependencia gráfica adjunta para HRI_SWM.h:

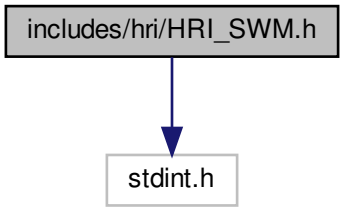
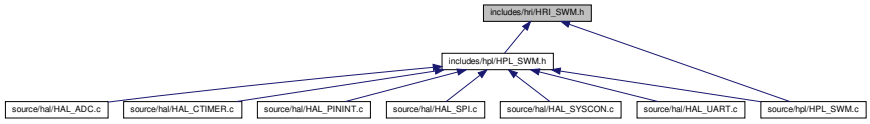


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct SWM_PINASSIGN0_reg_t
- struct SWM_PINASSIGN1_reg_t
- struct SWM_PINASSIGN2_reg_t
- struct SWM_PINASSIGN3_reg_t
- struct SWM_PINASSIGN4_reg_t
- struct SWM_PINASSIGN5_reg_t
- struct SWM_PINASSIGN6_reg_t
- struct SWM_PINASSIGN7_reg_t
- struct SWM_PINASSIGN8_reg_t
- struct SWM_PINASSIGN9_reg_t
- struct SWM_PINASSIGN10_reg_t

- struct SWM_PINASSIGN11_reg_t
- struct SWM_PINASSIGN12_reg_t
- struct SWM_PINASSIGN13_reg_t
- struct SWM_PINASSIGN14_reg_t
- struct SWM_PINENABLE0_reg_t
- struct SWM_PINENABLE1_reg_t
- struct SWM_per_t

defines

- #define SWM_BASE 0x4000C000

  Base del eriferico SWM.
- #define **PINENABLE_XTALIN_ON** 0
- #define **PINENABLE_XTALIN_OFF** 1
- #define **PINENABLE_XTALOUT_ON** 0
- #define **PINENABLE_XTALOUT_OFF** 1

7.36.1. Descripción detallada

Definiciones a nivel de registros del modulo SWM (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.36.2. Documentación de las estructuras de datos

7.36.2.1. struct SWM_PINASSIGN0_reg_t

Campos de datos

uint32_t	U0_TXD_O: 8	
uint32_t	U0_RXD_I: 8	
uint32_t	U0_RTS_O: 8	
uint32_t	U0_CTS_I: 8	

7.36.2.2. struct SWM_PINASSIGN1_reg_t

Campos de datos

uint32_t	U0_SCLK_IO: 8	
uint32_t	U1_TXD_O: 8	
uint32_t	U1_RXD_I: 8	
uint32_t	U1_RTS_O: 8	

7.36.2.3. struct SWM_PINASSIGN2_reg_t

Campos de datos

uint32_t	U1_CTS_I: 8	
uint32_t	U1_SCLK_IO: 8	
uint32_t	U2_TXD_O: 8	
uint32_t	U2_RXD_I: 8	

7.36.2.4. struct SWM_PINASSIGN3_reg_t

Campos de datos

uint32_t	U2_RTS_O: 8	
uint32_t	U2_CTS_I: 8	
uint32_t	U2_SCLK_IO: 8	
uint32_t	SPI0_SCK_IO: 8	

7.36.2.5. struct SWM_PINASSIGN4_reg_t

Campos de datos

uint32_t	SPI0_MOSI_IO: 8	
uint32_t	SPI0_MISO_IO: 8	
uint32_t	SPI0_SSEL0_IO: 8	
uint32_t	SPI0_SSEL1_IO: 8	

7.36.2.6. struct SWM_PINASSIGN5_reg_t

Campos de datos

uint32_t	SPI0_SSEL2_IO: 8	
uint32_t	SPI0_SSEL3_IO: 8	
uint32_t	SPI1_SCK_IO: 8	
uint32_t	SPI1_MOSI_IO: 8	

7.36.2.7. struct SWM_PINASSIGN6_reg_t

Campos de datos

uint32_t	SPI1_MISO_IO: 8	
uint32_t	SPI1_SSEL0_IO: 8	
uint32_t	SPI1_SSEL1_IO: 8	
uint32_t	SCT0_GPIO_IN_A_I: 8	

7.36.2.8. struct SWM_PINASSIGN7_reg_t

Campos de datos

uint32_t	SCT0_GPIO_IN_B_I: 8	
uint32_t	SCT0_GPIO_IN_C_I: 8	
uint32_t	SCT0_GPIO_IN_D_I: 8	
uint32_t	SCT_OUT0_O: 8	

7.36.2.9. struct SWM_PINASSIGN8_reg_t

Campos de datos

uint32_t	SCT_OUT1_O: 8	
uint32_t	SCT_OUT2_O: 8	
uint32_t	SCT_OUT3_O: 8	
uint32_t	SCT_OUT4_O: 8	

7.36.2.10. struct SWM_PINASSIGN9_reg_t

Campos de datos

uint32_t	SCT_OUT5_O: 8	
uint32_t	SCT_OUT6_O: 8	
uint32_t	I2C1_SDA_IO: 8	
uint32_t	I2C1_SCL_IO: 8	

7.36.2.11. struct SWM_PINASSIGN10_reg_t

Campos de datos

uint32_t	I2C2_SDA_IO: 8	
uint32_t	I2C2_SCL_IO: 8	
uint32_t	I2C3_SDA_IO: 8	
uint32_t	I2C3_SCL_IO: 8	

7.36.2.12. struct SWM_PINASSIGN11_reg_t

Campos de datos

uint32_t	COMP0_OUT_O: 8	
uint32_t	CLKOUT_O: 8	
uint32_t	GPIO_INT_BMAT_O: 8	
uint32_t	UART3_TXD: 8	

7.36.2.13. struct SWM_PINASSIGN12_reg_t

Campos de datos

uint32_t	UART3_RXD: 8	
uint32_t	UART3_SCLK: 8	
uint32_t	UART4_TXD: 8	
uint32_t	UART4_RXD: 8	

7.36.2.14. struct SWM_PINASSIGN13_reg_t

Campos de datos

uint32_t	UART4_SCLK: 8	
uint32_t	T0_MAT0: 8	
uint32_t	T0_MAT1: 8	
uint32_t	T0_MAT2: 8	

7.36.2.15. struct SWM_PINASSIGN14_reg_t

Campos de datos

uint32_t	T0_MAT3: 8	
uint32_t	T0_CAP0: 8	
uint32_t	T0_CAP1: 8	
uint32_t	T0_CAP2: 8	

7.36.2.16. struct SWM_PINENABLE0_reg_t

Campos de datos

uint32_t	ACMP_I1: 1	
uint32_t	ACMP_I2: 1	
uint32_t	ACMP_I3: 1	
uint32_t	ACMP_I4: 1	
uint32_t	ACMP_I5: 1	
uint32_t	SWCLK: 1	
uint32_t	SWDIO: 1	
uint32_t	XTALIN: 1	
uint32_t	XTALOUT: 1	
uint32_t	RESETN: 1	

Campos de datos

uint32_t	CLKIN: 1	
uint32_t	VDDCMP: 1	
uint32_t	I2C0_SDA: 1	
uint32_t	I2C0_SCL: 1	
uint32_t	ADC_0: 1	
uint32_t	ADC_1: 1	
uint32_t	ADC_2: 1	
uint32_t	ADC_3: 1	
uint32_t	ADC_4: 1	
uint32_t	ADC_5: 1	
uint32_t	ADC_6: 1	
uint32_t	ADC_7: 1	
uint32_t	ADC_8: 1	
uint32_t	ADC_9: 1	
uint32_t	ADC_10: 1	
uint32_t	ADC_11: 1	
uint32_t	DACOUT0: 1	
uint32_t	DACOUT1: 1	
uint32_t	CAPT_X0: 1	
uint32_t	CAPT_X1: 1	
uint32_t	CAPT_X2: 1	
uint32_t	CAPT_X3: 1	

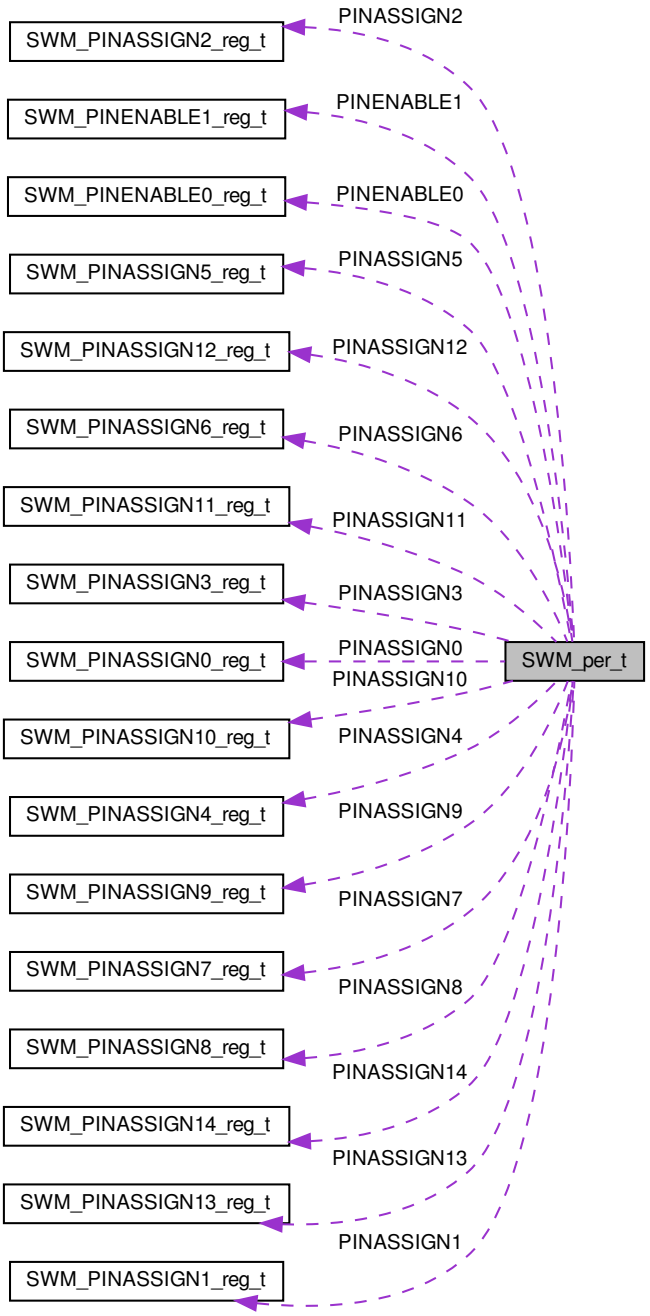
7.36.2.17. struct SWM_PINENABLE1_reg_t

Campos de datos

uint32_t	CAPT_X4: 1	
uint32_t	CAPT_X5: 1	
uint32_t	CAPT_X6: 1	
uint32_t	CAPT_X7: 1	
uint32_t	CAPT_X8: 1	
uint32_t	CAPT_YL: 1	
uint32_t	CAPT_YH: 1	
uint32_t	__pad0__: 26	

7.36.2.18. struct SWM_per_t

Diagrama de colaboración para SWM_per_t:



Campos de datos

SWM_PINASSIGN0_reg_t	PINASSIGN0	
SWM_PINASSIGN1_reg_t	PINASSIGN1	
SWM_PINASSIGN2_reg_t	PINASSIGN2	

Campos de datos

SWM_PINASSIGN3_reg_t	PINASSIGN3	
SWM_PINASSIGN4_reg_t	PINASSIGN4	
SWM_PINASSIGN5_reg_t	PINASSIGN5	
SWM_PINASSIGN6_reg_t	PINASSIGN6	
SWM_PINASSIGN7_reg_t	PINASSIGN7	
SWM_PINASSIGN8_reg_t	PINASSIGN8	
SWM_PINASSIGN9_reg_t	PINASSIGN9	
SWM_PINASSIGN10_reg_t	PINASSIGN10	
SWM_PINASSIGN11_reg_t	PINASSIGN11	
SWM_PINASSIGN12_reg_t	PINASSIGN12	
SWM_PINASSIGN13_reg_t	PINASSIGN13	
SWM_PINASSIGN14_reg_t	PINASSIGN14	
const uint32_t	RESERVED[(0x188 - 4)/4]	
SWM_PINENABLE0_reg_t	PINENABLE0	
SWM_PINENABLE1_reg_t	PINENABLE1	

7.37. Referencia del Archivo includes/hri/HRI_SYSCON.h

Definiciones a nivel de registros del modulo SYSCON (LPC845)

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

Dependencia gráfica adjunta para HRI_SYSCON.h:

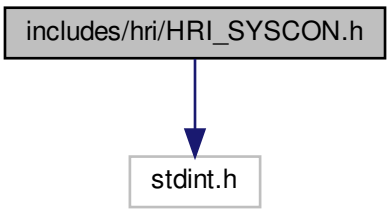
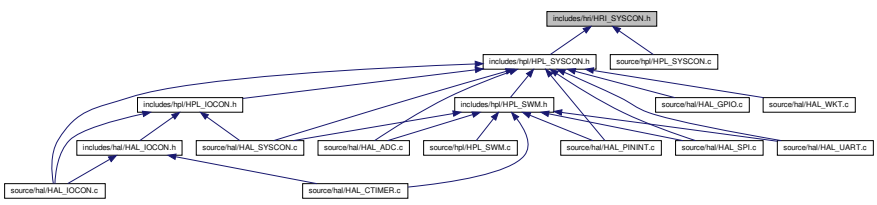


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct SYSCON_RESERVED_reg_t
- struct SYSCON_SYSMEMREMAP_reg_t
- struct SYSCON_SYSPLLCTRL_reg_t
- struct SYSCON_SYSPLLSTAT_reg_t
- struct SYSCON_SYSOSCCTRL_reg_t
- struct SYSCON_WDTOSCCTRL_reg_t
- struct SYSCON_FROOSCCTRL_reg_t
- struct SYSCON_FRODIRECTCLKUEN_reg_t
- struct SYSCON_SYSRSTSTAT_reg_t
- struct SYSCON_SYSPLLCLKSEL_reg_t
- struct SYSCON_SYSPLLCLKUEN_reg_t
- struct SYSCON_MAINCLKPLLSEL_reg_t
- struct SYSCON_MAINCLKPLLUEN_reg_t
- struct SYSCON_MAINCLKSEL_reg_t
- struct SYSCON_MAINCLKUEN_reg_t
- struct SYSCON_SYSAHBCLKDIV_reg_t
- struct SYSCON_CAPTCLKSEL_reg_t
- struct SYSCON_ADCCLKSEL_reg_t
- struct SYSCON_ADCCLKDIV_reg_t
- struct SYSCON_SCTCLKSEL_reg_t
- struct SYSCON_SCTCLKDIV_reg_t
- struct SYSCON_EXTCLKSEL_reg_t
- struct SYSCON_SYSAHBCLKCTRL0_reg_t
- struct SYSCON_SYSAHBCLKCTRL1_reg_t
- struct SYSCON_PRESETCTRL0_reg_t
- struct SYSCON_PRESETCTRL1_reg_t
- struct SYSCON_PERCLKSEL_reg_t
- struct SYSCON_FRGDIV_reg_t
- struct SYSCON_FRGMULT_reg_t
- struct SYSCON_FRGCLKSEL_reg_t
- struct SYSCON_CLKOUTSEL_reg_t
- struct SYSCON_CLKOUTDIV_reg_t
- struct SYSCON_EXTTRACECMD_reg_t
- struct SYSCON_PIOPORCAP_reg_t
- struct SYSCON_IOCONCLKDIV_reg_t
- struct SYSCON_BODCTRL_reg_t
- struct SYSCON_SYSTCKCAL_reg_t
- struct SYSCON_IRQLATENCY_reg_t
- struct SYSCON_NMISRC_reg_t
- struct SYSCON_PINTSEL_reg_t
- struct SYSCON_STARTERP0_reg_t
- struct SYSCON_STARTERP1_reg_t
- struct SYSCON_PDSLEEPCFG_reg_t
- struct SYSCON_PDAWAKECFG_reg_t
- struct SYSCON_PDRUNCFG_reg_t
- struct SYSCON_DEVICE_ID_reg_t
- struct SYSCON_per_t

defines

- #define **SYSCON_BASE** 0x40048000
- #define **SYSCON_PER_CLK_SOURCE_FRO** 0
- #define **SYSCON_PER_CLK_SOURCE_MAIN** 1
- #define **SYSCON_PER_CLK_SOURCE_FRG0** 2
- #define **SYSCON_PER_CLK_SOURCE_FRG1** 3
- #define **SYSCON_PER_CLK_SOURCE_FRO_DIV** 4
- #define **SYSCON_PER_CLK_SOURCE_NONE** 7

7.37.1. Descripción detallada

Definiciones a nivel de registros del modulo SYSCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.37.2. Documentación de las estructuras de datos

7.37.2.1. struct SYSCON_RESERVED_reg_t

Campos de datos

uint32_t	RESERVED	

7.37.2.2. struct SYSCON_SYSMEMREMAP_reg_t

Campos de datos

uint32_t	MAP: 2	
uint32_t	__pad0__: 30	

7.37.2.3. struct SYSCON_SYSPLLCTRL_reg_t

Campos de datos

uint32_t	MSEL: 4	
uint32_t	PSEL: 2	

Campos de datos

uint32_t	__pad0__: 26	

7.37.2.4. struct SYSCON_SYSPLLSTAT_reg_t

Campos de datos

uint32_t	LOCK: 1	
uint32_t	__pad0__: 31	

7.37.2.5. struct SYSCON_SYSOSCCTRL_reg_t

Campos de datos

uint32_t	BYPASS: 1	
uint32_t	FREQRANGE: 1	
uint32_t	__pad0__: 30	

7.37.2.6. struct SYSCON_WDTOSCCTRL_reg_t

Campos de datos

uint32_t	DIVSEL: 5	
uint32_t	FREQSEL: 4	
uint32_t	__pad0__: 23	

7.37.2.7. struct SYSCON_FROOSCCTRL_reg_t

Campos de datos

uint32_t	__pad0__: 17	
uint32_t	FRO_DIRECT: 1	
uint32_t	__pad1__: 14	

7.37.2.8. struct SYSCON_FRODIRECTCLKUEN_reg_t

Campos de datos

uint32_t	ENA: 1	
uint32_t	__pad0__: 31	

7.37.2.9. struct SYSCON_SYSRSTSTAT_reg_t

Campos de datos

uint32_t	POR: 1	
uint32_t	EXTRST: 1	
uint32_t	WDT: 1	
uint32_t	BOD: 1	
uint32_t	SYSRST: 1	
uint32_t	__pad0__: 27	

7.37.2.10. struct SYSCON_SYSPLLCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 2	
uint32_t	__pad0__: 30	

7.37.2.11. struct SYSCON_SYSPLLCLKUEN_reg_t

Campos de datos

uint32_t	ENA: 1	
uint32_t	__pad0__: 31	

7.37.2.12. struct SYSCON_MAINCLKPLLSEL_reg_t

Campos de datos

uint32_t	SEL: 2	
uint32_t	__pad0__: 30	

7.37.2.13. struct SYSCON_MAINCLKPLLUEN_reg_t

Campos de datos

uint32_t	ENA: 1	
uint32_t	__pad0__: 31	

7.37.2.14. struct SYSCON_MAINCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 2	
uint32_t	__pad0__: 30	

7.37.2.15. struct SYSCON_MAINCLKUEN_reg_t

Campos de datos

uint32_t	ENA: 1	
uint32_t	__pad0__: 31	

7.37.2.16. struct SYSCON_SYSAHBCLKDIV_reg_t

Campos de datos

uint32_t	DIV: 8	
uint32_t	__pad0__: 24	

7.37.2.17. struct SYSCON_CAPTCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 3	
uint32_t	__pad0__: 29	

7.37.2.18. struct SYSCON_ADCCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 2	
uint32_t	__pad0__: 30	

7.37.2.19. struct SYSCON_ADCCLKDIV_reg_t

Campos de datos

uint32_t	DIV: 8	
uint32_t	__pad0__: 24	

7.37.2.20. struct SYSCON_SCTCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 2	
uint32_t	__pad0__: 30	

7.37.2.21. struct SYSCON_SCTCLKDIV_reg_t

Campos de datos

uint32_t	DIV: 8	
uint32_t	__pad0__: 24	

7.37.2.22. struct SYSCON_EXTCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 1	
uint32_t	__pad0__: 31	

7.37.2.23. struct SYSCON_SYSAHBCLKCTRL0_reg_t

Campos de datos

uint32_t	SYS: 1	
uint32_t	ROM: 1	
uint32_t	RAM0_1: 1	
uint32_t	__pad0__: 1	
uint32_t	FLASH: 1	
uint32_t	I2C0: 1	
uint32_t	GPIO0: 1	
uint32_t	SWM: 1	
uint32_t	SCT: 1	
uint32_t	WKT: 1	
uint32_t	MRT: 1	
uint32_t	SPI0: 1	
uint32_t	SPI1: 1	
uint32_t	CRC: 1	
uint32_t	UART0: 1	
uint32_t	UART1: 1	
uint32_t	UART2: 1	
uint32_t	WWDT: 1	
uint32_t	IOCON: 1	
uint32_t	ACMP: 1	
uint32_t	GPIO1: 1	
uint32_t	I2C1: 1	
uint32_t	I2C2: 1	
uint32_t	I2C3: 1	
uint32_t	ADC: 1	
uint32_t	CTIMER0: 1	
uint32_t	MTB: 1	
uint32_t	DAC0: 1	
uint32_t	GPIO_INT: 1	
uint32_t	DMA: 1	
uint32_t	UART3: 1	
uint32_t	UART4: 1	

7.37.2.24. struct SYSCON_SYSAHBCLKCTRL1_reg_t

Campos de datos

uint32_t	CAPT: 1	

Campos de datos

uint32_t	DAC1: 1	
uint32_t	__pad0__: 30	

7.37.2.25. struct SYSCON_PRESETCTRL0_reg_t

Campos de datos

uint32_t	__pad0__: 4	
uint32_t	FLASH_RST_N: 1	
uint32_t	I2C0_RST_N: 1	
uint32_t	GPIO0_RST_N: 1	
uint32_t	SWM_RST_N: 1	
uint32_t	SCT_RST_N: 1	
uint32_t	WKT_RST_N: 1	
uint32_t	MRT_RST_N: 1	
uint32_t	SPI0_RST_N: 1	
uint32_t	SPI1_RST_N: 1	
uint32_t	CRC_RST_N: 1	
uint32_t	UART0_RST_N: 1	
uint32_t	UART1_RST_N: 1	
uint32_t	UART2_RST_N: 1	
uint32_t	__pad1__: 1	
uint32_t	IOCON_RST_N: 1	
uint32_t	ACMP_RST_N: 1	
uint32_t	GPIO1_RST_N: 1	
uint32_t	I2C1_RST_N: 1	
uint32_t	I2C2_RST_N: 1	
uint32_t	I2C3_RST_N: 1	
uint32_t	ADC_RST_N: 1	
uint32_t	CTIMER0_RST_N: 1	
uint32_t	__pad2__: 1	
uint32_t	DAC0_RST_N: 1	
uint32_t	GPIOINT_RST_N: 1	
uint32_t	DMA_RST_N: 1	
uint32_t	UART3_RST_N: 1	
uint32_t	UART4_RST_N: 1	

7.37.2.26. struct SYSCON_PRESETCTRL1_reg_t

Campos de datos

uint32_t	CAPT_RST_N: 1	
uint32_t	DAC1_RST_N: 1	
uint32_t	__pad0__: 1	
uint32_t	FRG0_RST_N: 1	
uint32_t	FRG1_RST_N: 1	
uint32_t	__pad1__: 27	

7.37.2.27. struct SYSCON_PERCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 3	
uint32_t	__pad0__: 29	

7.37.2.28. struct SYSCON_FRGDIV_reg_t

Campos de datos

uint32_t	DIV: 8	
uint32_t	__pad0__: 24	

7.37.2.29. struct SYSCON_FRGMULT_reg_t

Campos de datos

uint32_t	MULT: 8	
uint32_t	__pad0__: 24	

7.37.2.30. struct SYSCON_FRGCLKSEL_reg_t

Campos de datos

uint32_t	SEL: 2	
uint32_t	__pad0__: 30	

7.37.2.31. struct SYSCON_CLKOUTSEL_reg_t

Campos de datos

uint32_t	SEL: 3	
uint32_t	__pad0__: 29	

7.37.2.32. struct SYSCON_CLKOUTDIV_reg_t

Campos de datos

uint32_t	DIV: 8	
uint32_t	__pad0__: 24	

7.37.2.33. struct SYSCON_EXTTRACECMD_reg_t

Campos de datos

uint32_t	START: 1	
uint32_t	STOP: 1	
uint32_t	__pad0__: 30	

7.37.2.34. struct SYSCON_PIOPORCAP_reg_t

Campos de datos

uint32_t	PIOSTAT	

7.37.2.35. struct SYSCON_IOCONCLKDIV_reg_t

Campos de datos

uint32_t	DIV: 8	
uint32_t	__pad0__: 24	

7.37.2.36. struct SYSCON_BODCTRL_reg_t

Campos de datos

uint32_t	BODRSTLEV: 2	
uint32_t	BODINTVAL: 2	
uint32_t	BODRSTENA: 1	
uint32_t	__pad0__: 27	

7.37.2.37. struct SYSCON_SYSTCKCAL_reg_t

Campos de datos

uint32_t	CAL: 26	
uint32_t	__pad0__: 6	

7.37.2.38. struct SYSCON_IRQLATENCY_reg_t

Campos de datos

uint32_t	LATENCY: 8	
uint32_t	__pad0__: 24	

7.37.2.39. struct SYSCON_NMISRC_reg_t

Campos de datos

uint32_t	IRQN: 5	
uint32_t	__pad0__: 26	
uint32_t	NMIEN: 1	

7.37.2.40. struct SYSCON_PINTSEL_reg_t

Campos de datos

uint32_t	INTPIN: 6	
uint32_t	__pad0__: 26	

7.37.2.41. struct SYSCON_STARTERP0_reg_t

Campos de datos

uint32_t	PINT0: 1	
uint32_t	PINT1: 1	
uint32_t	PINT2: 1	
uint32_t	PINT3: 1	
uint32_t	PINT4: 1	
uint32_t	PINT5: 1	
uint32_t	PINT6: 1	
uint32_t	PINT7: 1	
uint32_t	__pad0__: 24	

7.37.2.42. struct SYSCON_STARTERP1_reg_t

Campos de datos

uint32_t	SPI0: 1	
uint32_t	SPI1: 1	
uint32_t	__pad0__: 1	
uint32_t	USART0: 1	
uint32_t	USART1: 1	
uint32_t	USART2: 1	
uint32_t	__pad1__: 1	
uint32_t	I2C1: 1	
uint32_t	I2C0: 1	
uint32_t	__pad2__: 2	
uint32_t	CAP_TOUCH: 1	
uint32_t	WWDT: 1	
uint32_t	BOD: 1	
uint32_t	__pad3__: 1	
uint32_t	WKT: 1	
uint32_t	__pad4__: 5	
uint32_t	I2C2: 1	

Campos de datos

uint32_t	I2C3: 1	
uint32_t	__pad5__: 7	
uint32_t	USART3: 1	
uint32_t	USART4: 1	

7.37.2.43. struct SYSCON_PDSLEEPCFG_reg_t

Campos de datos

uint32_t	__pad0__: 3	
uint32_t	BOD_PD: 1	
uint32_t	__pad1__: 2	
uint32_t	WDTOSC_PD: 1	
uint32_t	__pad2__: 25	

7.37.2.44. struct SYSCON_PDAWAKECFG_reg_t

Campos de datos

uint32_t	FRO_OUT_PD: 1	
uint32_t	FRO_PD: 1	
uint32_t	FLASH_PD: 1	
uint32_t	BOD_PD: 1	
uint32_t	ADC_PD: 1	
uint32_t	SYSOSC_PD: 1	
uint32_t	WDTOSC_PD: 1	
uint32_t	SYSPLL_PD: 1	
uint32_t	__pad0__: 2	
uint32_t	VREF2_PD: 1	
uint32_t	__pad1__: 2	
uint32_t	DAC0: 1	
uint32_t	DAC1: 1	
uint32_t	ACMP: 1	
uint32_t	__pad2__: 16	

7.37.2.45. struct SYSCON_PDRUNCFG_reg_t

Campos de datos

uint32_t	FROOUT_PD: 1	
uint32_t	FRO_PD: 1	
uint32_t	FLASH_PD: 1	
uint32_t	BOD_PD: 1	
uint32_t	ADC_PD: 1	
uint32_t	SYSOSC_PD: 1	
uint32_t	WDTOSC_PD: 1	

Campos de datos

uint32_t	SYSPLL_PD: 1	
uint32_t	__pad0__: 5	
uint32_t	DAC0: 1	
uint32_t	DAC1: 1	
uint32_t	ACMP: 1	
uint32_t	__pad1__: 16	

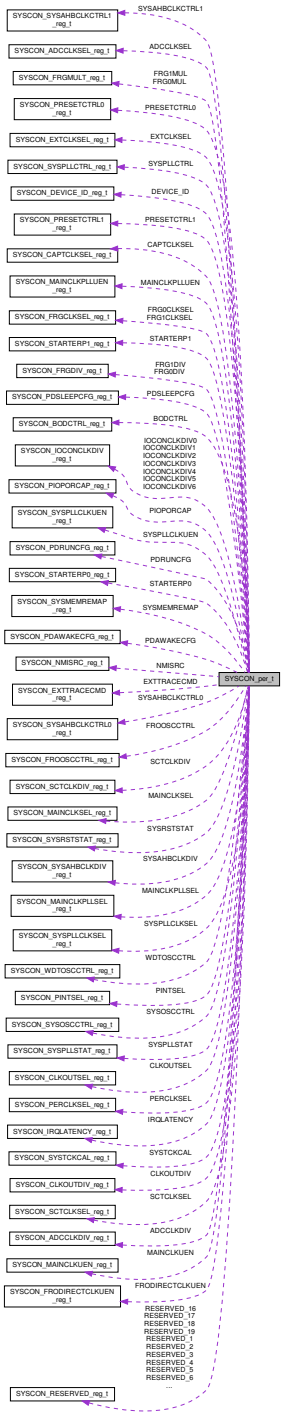
7.37.2.46. struct SYSCON_DEVICE_ID_reg_t

Campos de datos

uint32_t	DEVICE_ID	

7.37.2.47. struct SYSCON_per_t

Diagrama de colaboración para SYSCON_per_t:



Campos de datos

SYSCON_SYSMEMREMAP_reg_t	SYSMEMREMAP	
const SYSCON_RESERVED_reg_t	RESERVED_1	
SYSCON_SYSPLLCTRL_reg_t	SYSPLLCTRL	

Campos de datos

const SYSCON_SYSPLLSTAT_reg_t	SYSPLLSTAT	
const SYSCON_RESERVED_reg_t	RESERVED_2	
const SYSCON_RESERVED_reg_t	RESERVED_3	
const SYSCON_RESERVED_reg_t	RESERVED_4	
const SYSCON_RESERVED_reg_t	RESERVED_5	
SYSCON_SYSOSCCTRL_reg_t	SYSOSCCTRL	
SYSCON_WDTOSCCTRL_reg_t	WDTOSCCTRL	
SYSCON_FROOSCCTRL_reg_t	FROOSCCTRL	
const SYSCON_RESERVED_reg_t	RESERVED_6	
SYSCON_FRODIRECTCLKUEN_reg_t	FRODIRECTCLKUEN	
const SYSCON_RESERVED_reg_t	RESERVED_7	
SYSCON_SYSRSTSTAT_reg_t	SYSRSTSTAT	
const SYSCON_RESERVED_reg_t	RESERVED_8	
SYSCON_SYSPLLCLKSEL_reg_t	SYSPLLCLKSEL	
SYSCON_SYSPLLCLKUEN_reg_t	SYSPLLCLKUEN	
SYSCON_MAINCLKPLLSEL_reg_t	MAINCLKPLLSEL	
SYSCON_MAINCLKPLLUEN_reg_t	MAINCLKPLLUEN	
SYSCON_MAINCLKSEL_reg_t	MAINCLKSEL	
SYSCON_MAINCLKUEN_reg_t	MAINCLKUEN	
SYSCON_SYSAHBCLKDIV_reg_t	SYSAHBCLKDIV	
const SYSCON_RESERVED_reg_t	RESERVED_9	
SYSCON_CAPTCLKSEL_reg_t	CAPTCLKSEL	
SYSCON_ADCCLKSEL_reg_t	ADCCLKSEL	
SYSCON_ADCCLKDIV_reg_t	ADCCLKDIV	
SYSCON_SCTCLKSEL_reg_t	SCTCLKSEL	
SYSCON_SCTCLKDIV_reg_t	SCTCLKDIV	
SYSCON_EXTCLKSEL_reg_t	EXTCLKSEL	
const SYSCON_RESERVED_reg_t	RESERVED_10	
const SYSCON_RESERVED_reg_t	RESERVED_11	
SYSCON_SYSAHBCLKCTRL0_reg_t	SYSAHBCLKCTRL0	
SYSCON_SYSAHBCLKCTRL1_reg_t	SYSAHBCLKCTRL1	
SYSCON_PRESETCTRL0_reg_t	PRESETCTRL0	
SYSCON_PRESETCTRL1_reg_t	PRESETCTRL1	
SYSCON_PERCLKSEL_reg_t	PERCLKSEL[11]	
const SYSCON_RESERVED_reg_t	RESERVED_12	
const SYSCON_RESERVED_reg_t	RESERVED_13	
const SYSCON_RESERVED_reg_t	RESERVED_14	
const SYSCON_RESERVED_reg_t	RESERVED_15	
const SYSCON_RESERVED_reg_t	RESERVED_16	
SYSCON_FRGDIV_reg_t	FRG0DIV	
SYSCON_FRGMULT_reg_t	FRG0MUL	
SYSCON_FRGCLKSEL_reg_t	FRG0CLKSEL	
const SYSCON_RESERVED_reg_t	RESERVED_17	
SYSCON_FRGDIV_reg_t	FRG1DIV	
SYSCON_FRGMULT_reg_t	FRG1MUL	
SYSCON_FRGCLKSEL_reg_t	FRG1CLKSEL	

Campos de datos

const SYSCON_RESERVED_reg_t	RESERVED_18	
SYSCON_CLKOUTSEL_reg_t	CLKOUTSEL	
SYSCON_CLKOUTDIV_reg_t	CLKOUTDIV	
const SYSCON_RESERVED_reg_t	RESERVED_19	
SYSCON_EXTTRACECMD_reg_t	EXTTRACECMD	
const SYSCON_PIOPORCAP_reg_t	PIOPORCAP[2]	
const SYSCON_RESERVED_reg_t	RESERVED_20[0x2C/4]	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV6	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV5	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV4	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV3	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV2	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV1	
SYSCON_IOCONCLKDIV_reg_t	IOCONCLKDIV0	
SYSCON_BODCTRL_reg_t	BODCTRL	
SYSCON_SYSTCKCAL_reg_t	SYSTCKCAL	
const SYSCON_RESERVED_reg_t	RESERVED_21[0x18/4]	
SYSCON_IRQLATENCY_reg_t	IRQLATENCY	
SYSCON_NMISRC_reg_t	NMISRC	
SYSCON_PINTSEL_reg_t	PINTSEL[8]	
const SYSCON_RESERVED_reg_t	RESERVED_22[0x6C/4]	
SYSCON_STARTERP0_reg_t	STARTERP0	
const SYSCON_RESERVED_reg_t	RESERVED_23[0x0C/4]	
SYSCON_STARTERP1_reg_t	STARTERP1	
const SYSCON_RESERVED_reg_t	RESERVED_24[0x18/4]	
SYSCON_PDSLEEPCFG_reg_t	PDSLEEPCFG	
SYSCON_PDAWAKECFG_reg_t	PDAWAKECFG	
SYSCON_PDRUNCFG_reg_t	PDRUNCFG	
const SYSCON_RESERVED_reg_t	RESERVED_25[0x1BC/4]	
const SYSCON_DEVICE_ID_reg_t	DEVICE_ID	

7.38. Referencia del Archivo includes/hri/HRI_SYSTICK.h

Definiciones a nivel de registros del modulo SYSTICK (LPC845)

```
#include <stdint.h>
```
Dependencia gráfica adjunta para HRI_SYSTICK.h:

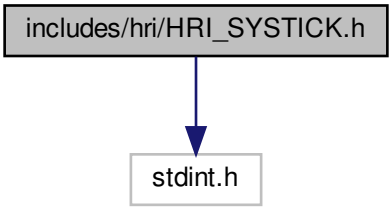
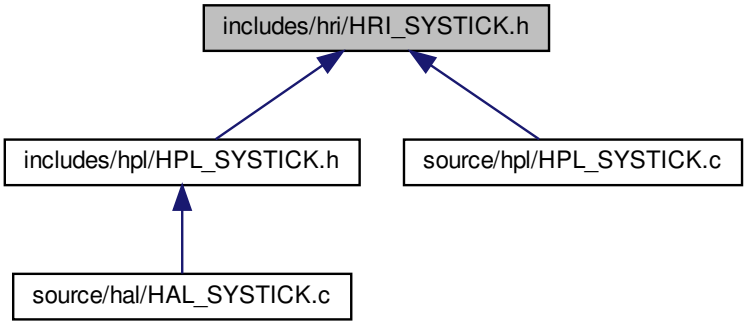


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct SYSTICK_RESERVED_reg_t
- struct SYSTICK_CSR_reg_t
- struct SYSTICK_RVR_reg_t
- struct SYSTICK_CVR_reg_t
- struct SYSTICK_CALIB_reg_t
- struct SYSTICK_reg_t

defines

- #define SYSTICK_BASE 0xE000E000

  Base del periferico SYSTICK.

7.38.1. Descripción detallada

Definiciones a nivel de registros del modulo SYSTICK (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.38.2. Documentación de las estructuras de datos

7.38.2.1. struct SYSTICK_RESERVED_reg_t

Campos de datos

uint32_t	RESERVED	

7.38.2.2. struct SYSTICK_CSR_reg_t

Campos de datos

uint32_t	ENABLE: 1	
uint32_t	TICKINT: 1	
uint32_t	CLKSOURCE: 1	
uint32_t	__pad0__: 13	
uint32_t	COUNTFLAG: 1	
uint32_t	__pad1__: 15	

7.38.2.3. struct SYSTICK_RVR_reg_t

Campos de datos

uint32_t	RELOAD: 23	
uint32_t	__pad0__: 9	

7.38.2.4. struct SYSTICK_CVR_reg_t

Campos de datos

uint32_t	CURRENT: 23	
uint32_t	__pad0__: 9	

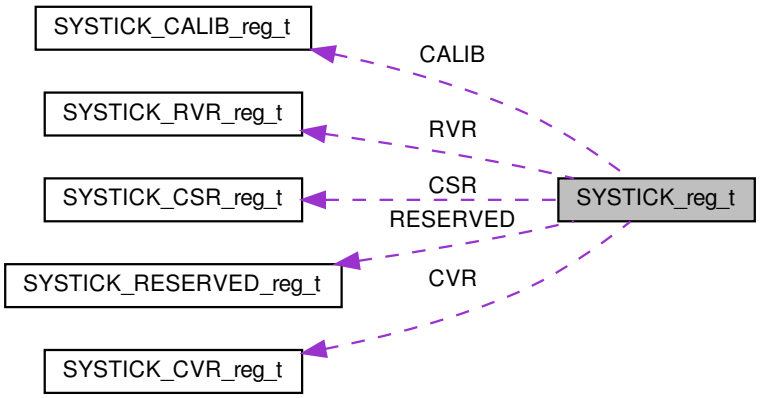
7.38.2.5. struct SYSTICK_CALIB_reg_t

Campos de datos

uint32_t	TENMS: 23	
uint32_t	__pad0__: 7	
uint32_t	SKEW: 1	
uint32_t	NOREF: 1	

7.38.2.6. struct SYSTICK_reg_t

Diagrama de colaboración para SYSTICK_reg_t:



Campos de datos

const SYSTICK_RESERVED_reg_t	RESERVED[4]	
SYSTICK_CSR_reg_t	CSR	
SYSTICK_RVR_reg_t	RVR	
SYSTICK_CVR_reg_t	CVR	
SYSTICK_CALIB_reg_t	CALIB	

7.39. Referencia del Archivo includes/hri/HRI_UART.h

Definiciones a nivel de registros del modulo UART (LPC845)

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

Dependencia gráfica adjunta para HRI_UART.h:

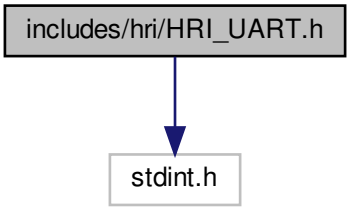
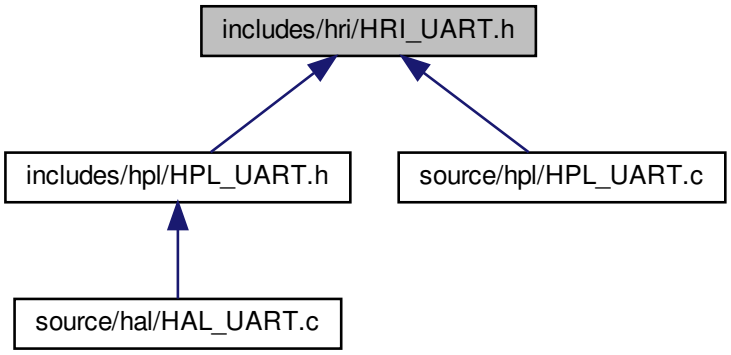


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct UART_CFG_reg_t
- struct UART_CTL_reg_t
- struct UART_STAT_reg_t
- struct UART_INTENSET_reg_t
- struct UART_INTENCLR_reg_t
- struct UART_RXDAT_reg_t
- struct UART_RXDATSTAT_reg_t
- struct UART_TXDAT_reg_t
- struct UART_BRG_reg_t
- struct UART_INTSTAT_reg_t
- struct UART_OSR_reg_t
- struct UART_ADDR_reg_t
- struct UART_per_t

defines

- #define **UART0_BASE** 0x40064000
- #define **UART1_BASE** 0x40068000
- #define **UART2_BASE** 0x4006C000
- #define **UART3_BASE** 0x40070000
- #define **UART4_BASE** 0x40074000

7.39.1. Descripción detallada

Definiciones a nivel de registros del modulo UART (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.39.2. Documentación de las estructuras de datos

7.39.2.1. struct UART_CFG_reg_t

Campos de datos

uint32_t	ENABLE: 1	
uint32_t	__pad0__: 1	
uint32_t	DATALEN: 2	
uint32_t	PARITYSEL: 2	
uint32_t	STOPLEN: 1	
uint32_t	__pad1__: 2	
uint32_t	CTSEN: 1	
uint32_t	__pad2__: 1	
uint32_t	SYNCEN: 1	
uint32_t	CLKPOL: 1	
uint32_t	__pad3__: 1	
uint32_t	SYNCMST: 1	
uint32_t	LOOP:1	
uint32_t	__pad4__: 2	
uint32_t	OETA: 1	
uint32_t	AUTOADDR: 1	
uint32_t	OESEL: 1	
uint32_t	OEPOL: 1	
uint32_t	RXPOL: 1	
uint32_t	TXPOL: 1	
uint32_t	__pad5__: 8	

7.39.2.2. struct UART_CTL_reg_t

Campos de datos

uint32_t	__pad0__: 1	
uint32_t	TXBRKEN: 1	
uint32_t	ADDRDET: 1	
uint32_t	__pad1__: 3	
uint32_t	TXDIS: 1	
uint32_t	__pad2__: 1	
uint32_t	CC: 1	
uint32_t	CLRCONRX: 1	
uint32_t	__pad3__: 6	
uint32_t	AUTOBAUD: 1	
uint32_t	__pad4__: 15	

7.39.2.3. struct UART_STAT_reg_t

Campos de datos

uint32_t	RXRDY: 1	
uint32_t	RXIDLE: 1	
uint32_t	TXRDY: 1	
uint32_t	TXIDLE: 1	
uint32_t	CTS: 1	
uint32_t	DELTACTS: 1	
uint32_t	TXDISSTAT: 1	
uint32_t	__pad0__: 1	
uint32_t	OVERRUNINT: 1	
uint32_t	__pad1__: 1	
uint32_t	RXBRK: 1	
uint32_t	DELTARXBRK: 1	
uint32_t	START: 1	
uint32_t	FRAMERRINT: 1	
uint32_t	PARITYERRINT: 1	
uint32_t	RXNOISEINT: 1	
uint32_t	ABERR: 1	
uint32_t	__pad2__: 14	

7.39.2.4. struct UART_INTENSET_reg_t

Campos de datos

uint32_t	RXRDYEN: 1	
uint32_t	__pad0__: 1	
uint32_t	TXRDYEN: 1	
uint32_t	TXIDLEEN: 1	
uint32_t	__pad1__: 1	
uint32_t	DELTACTSEN: 1	

Campos de datos

uint32_t	TXDISEN: 1	
uint32_t	__pad2__: 1	
uint32_t	OVERRUNEN: 1	
uint32_t	__pad3__: 2	
uint32_t	DELTARXBRKEN: 1	
uint32_t	STARTEN: 1	
uint32_t	FRAMERREN: 1	
uint32_t	PARITYERREN: 1	
uint32_t	RXNOISEEN: 1	
uint32_t	ABERREN: 1	
uint32_t	__pad4__: 14	

7.39.2.5. struct UART_INTENCLR_reg_t

Campos de datos

uint32_t	RXRDYCLR: 1	
uint32_t	__pad0__: 1	
uint32_t	TXRDYCLR: 1	
uint32_t	TXIDLECLR: 1	
uint32_t	__pad1__: 1	
uint32_t	DELTACTSCLR: 1	
uint32_t	TXDISCLR: 1	
uint32_t	__pad2__: 1	
uint32_t	OVERRUNCLR: 1	
uint32_t	__pad3__: 2	
uint32_t	DELTARXBRKCLR: 1	
uint32_t	STARTCLR: 1	
uint32_t	FRAMERRCLR: 1	
uint32_t	PARITYERRCLR: 1	
uint32_t	RXNOISECLR: 1	
uint32_t	ABERRCLR: 1	
uint32_t	__pad4__: 14	

7.39.2.6. struct UART_RXDAT_reg_t

Campos de datos

uint32_t	RXDAT: 9	
uint32_t	__pad0__: 23	

7.39.2.7. struct UART_RXDATSTAT_reg_t

Campos de datos

uint32_t	RXDAT: 9	

Campos de datos

uint32_t	__pad0__: 3	
uint32_t	FRAMERR: 1	
uint32_t	PARITYERR: 1	
uint32_t	RXNOISE: 1	
uint32_t	__pad1__: 17	

7.39.2.8. struct UART_TXDAT_reg_t

Campos de datos

uint32_t	TXDAT: 9	
uint32_t	__pad0__: 23	

7.39.2.9. struct UART_BRG_reg_t

Campos de datos

uint32_t	BRGVAL: 16	
uint32_t	__pad0__: 16	

7.39.2.10. struct UART_INTSTAT_reg_t

Campos de datos

uint32_t	RXRDY: 1	
uint32_t	__pad0__: 1	
uint32_t	TXRDY: 1	
uint32_t	TXIDLE: 1	
uint32_t	__pad1__: 1	
uint32_t	DELTACTS: 1	
uint32_t	TXDISINT: 1	
uint32_t	__pad2__: 1	
uint32_t	OVERRUNINT: 1	
uint32_t	__pad3__: 2	
uint32_t	DELTARXBRK: 1	
uint32_t	START: 1	
uint32_t	FRAMERRINT: 1	
uint32_t	PARITYERRINT: 1	
uint32_t	RXNOISEINT: 1	
uint32_t	ABERR: 1	
uint32_t	__pad4__: 15	

7.39.2.11. struct UART_OSR_reg_t

Campos de datos

uint32_t	OSRVAL: 4	
uint32_t	__pad0__: 28	

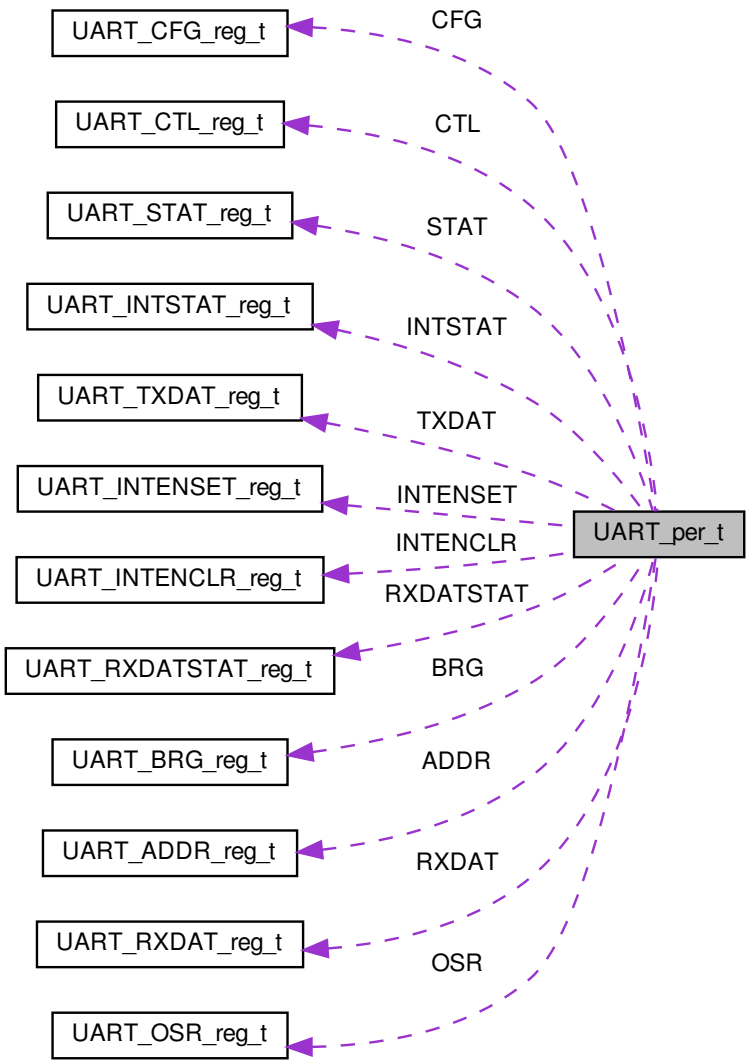
7.39.2.12. struct UART_ADDR_reg_t

Campos de datos

uint32_t	ADDRESS: 8	
uint32_t	__pad0__: 24	

7.39.2.13. struct UART_per_t

Diagrama de colaboración para UART_per_t:



Campos de datos

UART_CFG_reg_t	CFG	
UART_CTL_reg_t	CTL	
UART_STAT_reg_t	STAT	
UART_INTENSET_reg_t	INTENSET	
UART_INTENCLR_reg_t	INTENCLR	
const UART_RXDAT_reg_t	RXDAT	
const UART_RXDATSTAT_reg_t	RXDATSTAT	
UART_TXDAT_reg_t	TXDAT	
UART_BRG_reg_t	BRG	
const UART_INTSTAT_reg_t	INTSTAT	
UART_OSR_reg_t	OSR	
UART_ADDR_reg_t	ADDR	

7.40. Referencia del Archivo includes/hri/HRI_WKT.h

Definiciones a nivel de registros del periferico WKT (LPC845)

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

Dependencia gráfica adjunta para HRI_WKT.h:

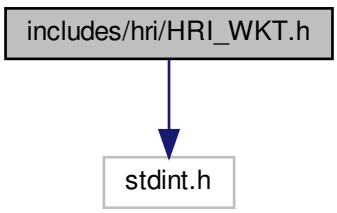
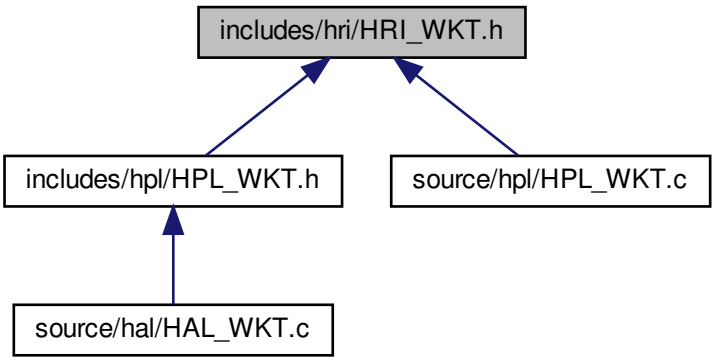


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Estructuras de datos

- struct WKT_CTRL_reg_t
- struct WKT_COUNT_reg_t
- struct WKT_per_t

defines

- #define **WKT_BASE** 0x40008000

7.40.1. Descripción detallada

Definiciones a nivel de registros del periferico WKT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.40.2. Documentación de las estructuras de datos

7.40.2.1. struct WKT_CTRL_reg_t

Campos de datos

uint32_t	CLKSEL: 1	
uint32_t	ALARMFLAG: 1	
uint32_t	CLEARCTR: 1	
uint32_t	SEL_EXTCLK: 1	
uint32_t	__pad0__: 28	

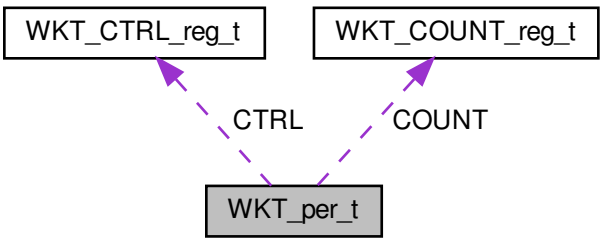
7.40.2.2. struct WKT_COUNT_reg_t

Campos de datos

uint32_t	VALUE	

7.40.2.3. struct WKT_per_t

Diagrama de colaboración para WKT_per_t:



Campos de datos

WKT_CTRL_reg_t	CTRL	
const uint32_t	RESERVED[2]	
WKT_COUNT_reg_t	COUNT	

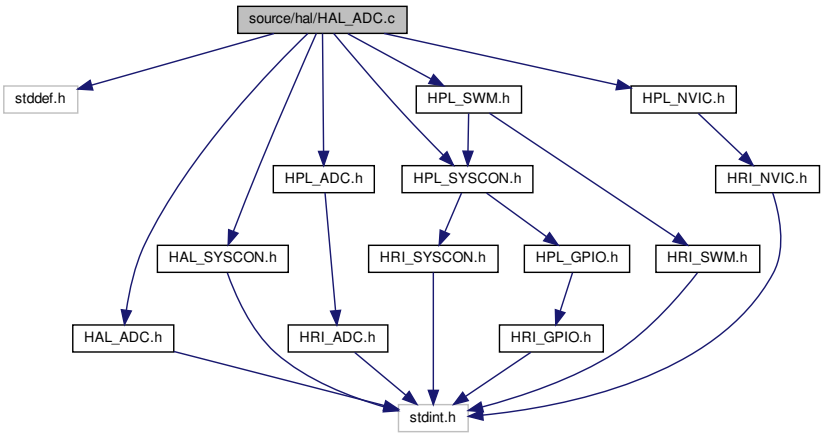
7.41. Referencia del Archivo source/hal/HAL_ADC.c

Funciones a nivel de aplicacion del periferico ADC (LPC845)

```
#include <stddef.h>
#include <HAL_ADC.h>
#include <HAL_SYSCON.h>
#include <HPL_ADC.h>
```

```
#include <HPL_SYSCON.h>
#include <HPL_SWM.h>
#include <HPL_NVIC.h>
```

Dependencia gráfica adjunta para HAL_ADC.c:



defines

- #define **ADC_MAX_FREQ_SYNC** ((uint32_t) 1.2e6)
- #define **ADC_MAX_FREQ_ASYNC** ((uint32_t) 0.6e6)
- #define **ADC_CYCLE_DELAY** (25)
- #define **ADC_CHANNEL_AMOUNT** (12)

Funciones

- static void dummy_irq_callback (void)

  Funcion dummy para usar como default para las interrupciones.
- void hal_adc_init_async_mode (uint32_t sample_freq, uint8_t div, hal_adc_clock_source_en clock_source, hal_adc_low_power_mode_en low_power)

  *Inicializar el ADC en modo **asincrónico**.*
- void hal_adc_init_sync_mode (uint32_t sample_freq, hal_adc_low_power_mode_en low_power)

  *Inicializar el ADC en modo **sincrónico**.*
- void hal_adc_deinit (void)

  De-inicialización del ADC.
- void hal_adc_config_sequence (hal_adc_sequence_sel_en sequence, const hal_adc_sequence_config_t *config)

  Configurar una secuencia de conversión.
- void hal_adc_enable_sequence (hal_adc_sequence_sel_en sequence)

  Habilitar una secuencia.
- void hal_adc_start_sequence (hal_adc_sequence_sel_en sequence)

  Disparar conversiones en una secuencia.
- hal_adc_sequence_result_en hal_adc_get_sequence_result (hal_adc_sequence_sel_en sequence, hal_adc_sequence_result_t *result)

  Obtener resultado de la secuencia.
- void ADC_SEQA_IRQHandler (void)

Funcion de interrupcion cuando termina la secuencia de conversion A del ADC.

- void ADC_SEQB_IRQHandler (void)

  Funcion de interrupcion cuando termina la secuencia de conversion B del ADC.
- void ADC_THCMP_IRQHandler (void)

  Funcion de interrupcion cuando se detecta alguna de las condiciones de threshold establecidas.
- void ADC_OVR_IRQHandler (void)

  Funcion de interrupcion cuando se detecta alguna de las condiciones de overrun.

Variables

- static void(∗ adc_seq_completed_callback [2])(void)
- static void(∗ adc_overrun_callback)(void) = dummy_irq_callback

  Callback cuando ocurre un overrun.
- static void(∗ adc_compare_callback)(void) = dummy_irq_callback

  Callbacks para las comparaciones de ADC.

7.41.1. Descripción detallada

Funciones a nivel de aplicacion del periferico ADC (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.41.2. Documentación de las variables

7.41.2.1. adc_seq_completed_callback

```
void(* adc_seq_completed_callback[2])(void)  [static]
```

Valor inicial:

```
=
{
    dummy_irq_callback,
    dummy_irq_callback
}
```

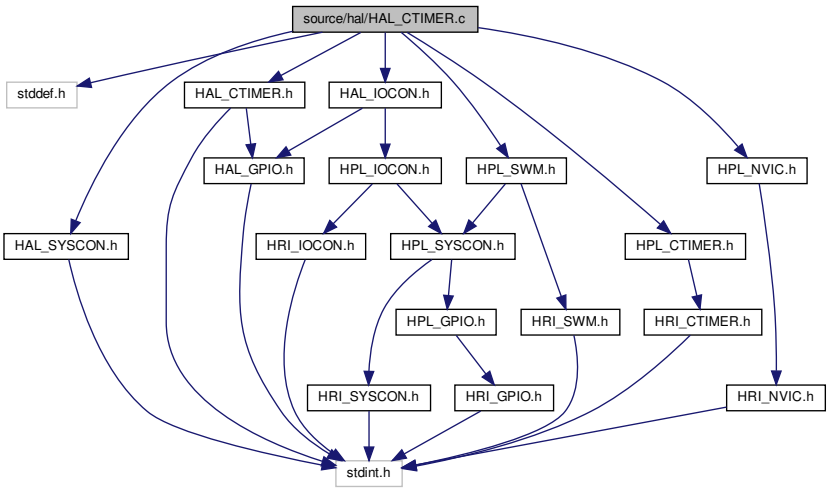
Callback cuando terminan las secuencias de conversion

7.42. Referencia del Archivo source/hal/HAL_CTIMER.c

Funciones a nivel de aplicacion del periferico CTIMER (LPC845)

```
#include <stddef.h>
#include <HAL_CTIMER.h>
#include <HAL_SYSCON.h>
#include <HAL_IOCON.h>
#include <HPL_CTIMER.h>
#include <HPL_NVIC.h>
#include <HPL_SWM.h>
```

Dependencia gráfica adjunta para HAL_CTIMER.c:



defines

- #define **MATCH_AMOUNT** 4
- #define **CAPTURE_AMOUNT** 4
- #define **PWM_CHANNELS** 3

Funciones

- static void dummy_irq (void)

  Funcion dummy para inicializar los callbacks de interrupcion.
- static uint32_t hal_ctimer_calc_match_value (uint32_t match_value_useg)

  Calcular el valor que debe ir en el registro de match a partir de un valor de useg deseado.
- void hal_ctimer_timer_mode_init (uint32_t clock_div)

  Inicializacion del periferico en modo timer.
- void hal_ctimer_timer_mode_config_match (hal_ctimer_match_sel_en match_sel, const hal_ctimer_match_config_t *match_config)

  Configurar un canal de match.
- void hal_ctimer_timer_mode_run (void)

  Habilitar el conteo del ctimer.
- void hal_ctimer_timer_mode_stop (void)

Inhabilitar el conteo del ctimer.

- void hal_ctimer_timer_mode_reset (void)

  Reiniciar el conteo del ctimer.
- void hal_ctimer_pwm_mode_init (const hal_ctimer_pwm_config_t *config)

  Inicializar el CTIMER en modo PWM.
- void hal_ctimer_pwm_mode_set_period (uint32_t period_useg)

  Actualizar el periodo en modo PWM.
- void hal_ctimer_pwm_mode_config_channel (hal_ctimer_pwm_channel_sel_en channel_sel, const hal_ctimer_pwm_channel_config_t *channel_config)

  Actualizar configuracion de algun canal de PWM.
- void CTIMER0_IRQHandler (void)

  Interrupcion de CTIMER.

Variables

- void(* match_callbacks [MATCH_AMOUNT])(void)

  Callbacks para interrupciones de match.
- void(* capture_callbacks [CAPTURE_AMOUNT])(void)

  Callbacks para interrupciones de capture.

7.42.1. Descripción detallada

Funciones a nivel de aplicacion del periferico CTIMER (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.42.2. Documentación de las funciones

7.42.2.1. hal_ctimer_calc_match_value()

```
static uint32_t hal_ctimer_calc_match_value (
            uint32_t match_value_useg )  [static]
```

Calcular el valor que debe ir en el registro de match a partir de un valor de useg deseado.

Devuelve

Valor que debe ir en el registro de match

7.42.2.2. hal_ctimer_timer_mode_init()

```
void hal_ctimer_timer_mode_init (
            uint32_t clock_div )
```

Inicializacion del periferico en modo timer.

Esta funcion no pone a correr el contador.

Parámetros

in	*clock_div*	Divisor del clock principal deseado (el valor efectivo es este valor + 1)

7.42.2.3. hal_ctimer_timer_mode_config_match()

```
void hal_ctimer_timer_mode_config_match (
            hal_ctimer_match_sel_en match_sel,
            const hal_ctimer_match_config_t * match_config )
```

Configurar un canal de match.

Parámetros

in	*match_sel*	Match a configurar
in	*match_config*	Configuracion deseada

7.42.2.4. hal_ctimer_pwm_mode_init()

```
void hal_ctimer_pwm_mode_init (
            const hal_ctimer_pwm_config_t * config )
```

Inicializar el CTIMER en modo PWM.

Parámetros

in	*config*	Configuracion deseada

7.42.2.5. hal_ctimer_pwm_mode_set_period()

```
void hal_ctimer_pwm_mode_set_period (
            uint32_t period_useg )
```

Actualizar el periodo en modo PWM.

Parámetros

in	*period_useg*	Nuevo periodo deseado en microsegundos

7.42.2.6. hal_ctimer_pwm_mode_config_channel()

```
void hal_ctimer_pwm_mode_config_channel (
            hal_ctimer_pwm_channel_sel_en channel_sel,
            const hal_ctimer_pwm_channel_config_t * channel_config )
```

Actualizar configuracion de algun canal de PWM.

Parámetros

in	*channel_sel*	Seleccion de canal a configurar
in	*channel_config*	Configuracion del canal de PWM

7.42.3. Documentación de las variables

7.42.3.1. match_callbacks

```
void(* match_callbacks[MATCH_AMOUNT])(void)
```

Valor inicial:

```
=
{
    dummy_irq,
    dummy_irq,
    dummy_irq,
    dummy_irq
}
```

Callbacks para interrupciones de match.

7.42.3.2. capture_callbacks

```
void(* capture_callbacks[CAPTURE_AMOUNT])(void)
```

Valor inicial:

```
=
{
    dummy_irq,
    dummy_irq,
    dummy_irq,
    dummy_irq
}
```

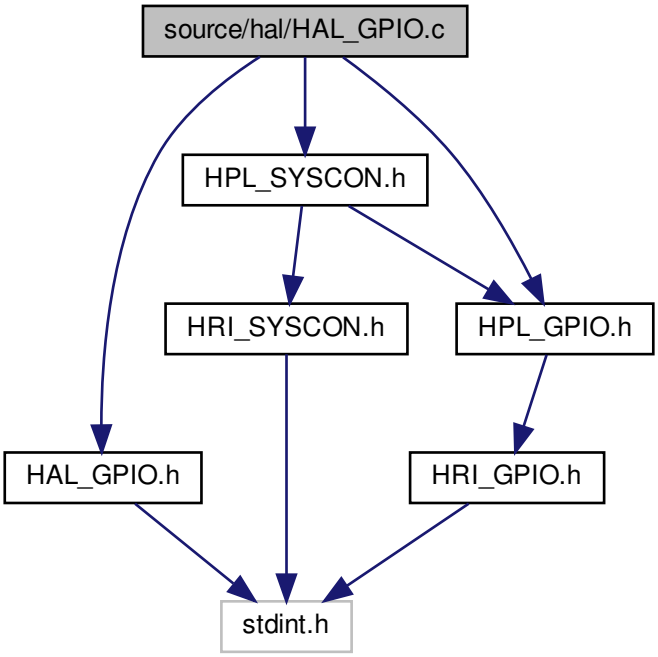
Callbacks para interrupciones de capture.

7.43. Referencia del Archivo source/hal/HAL_GPIO.c

Funciones a nivel de aplicacion del periferico GPIO (LPC845)

```
#include <HAL_GPIO.h>
#include <HPL_SYSCON.h>
#include <HPL_GPIO.h>
```

Dependencia gráfica adjunta para HAL_GPIO.c:



Funciones

- void hal_gpio_init (hal_gpio_port_en port)

  Inicializar un puerto.
- void hal_gpio_set_dir (hal_gpio_portpin_en portpin, hal_gpio_dir_en dir, uint8_t initial_state)

  Fijar direccion de una GPIO.
- void hal_gpio_set_pin (hal_gpio_portpin_en portpin)

  Fijar estado activo de una GPIO.
- void hal_gpio_clear_pin (hal_gpio_portpin_en portpin)

  Fijar estado inactivo de una GPIO.
- void hal_gpio_toggle_pin (hal_gpio_portpin_en portpin)

  Invertir estado de una GPIO.
- uint8_t hal_gpio_read_pin (hal_gpio_portpin_en portpin)

  Leer el estado de una GPIO.

7.43.1. Descripción detallada

Funciones a nivel de aplicacion del periferico GPIO (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.43.2. Documentación de las funciones

7.43.2.1. hal_gpio_init()

```
void hal_gpio_init (
            hal_gpio_port_en port )
```

Inicializar un puerto.

Parámetros

in	*port*	Puerto a inicializar

7.43.2.2. hal_gpio_set_dir()

```
void hal_gpio_set_dir (
            hal_gpio_portpin_en portpin,
            hal_gpio_dir_en dir,
            uint8_t initial_state )
```

Fijar direccion de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a configurar
in	*dir*	Direccion deseada
in	*initial_state*	Estado inicial (aplica para salidas nada mas)

7.43.2.3. hal_gpio_set_pin()

```
void hal_gpio_set_pin (
            hal_gpio_portpin_en portpin )
```

Fijar estado activo de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

7.43.2.4. hal_gpio_clear_pin()

```
void hal_gpio_clear_pin (
            hal_gpio_portpin_en portpin )
```

Fijar estado inactivo de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

7.43.2.5. hal_gpio_toggle_pin()

```
void hal_gpio_toggle_pin (
            hal_gpio_portpin_en portpin )
```

Invertir estado de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

7.43.2.6. hal_gpio_read_pin()

```
uint8_t hal_gpio_read_pin (
            hal_gpio_portpin_en portpin )
```

Leer el estado de una GPIO.

Parámetros

in	*portpin*	Numero de puerto/pin a accionar

Devuelve

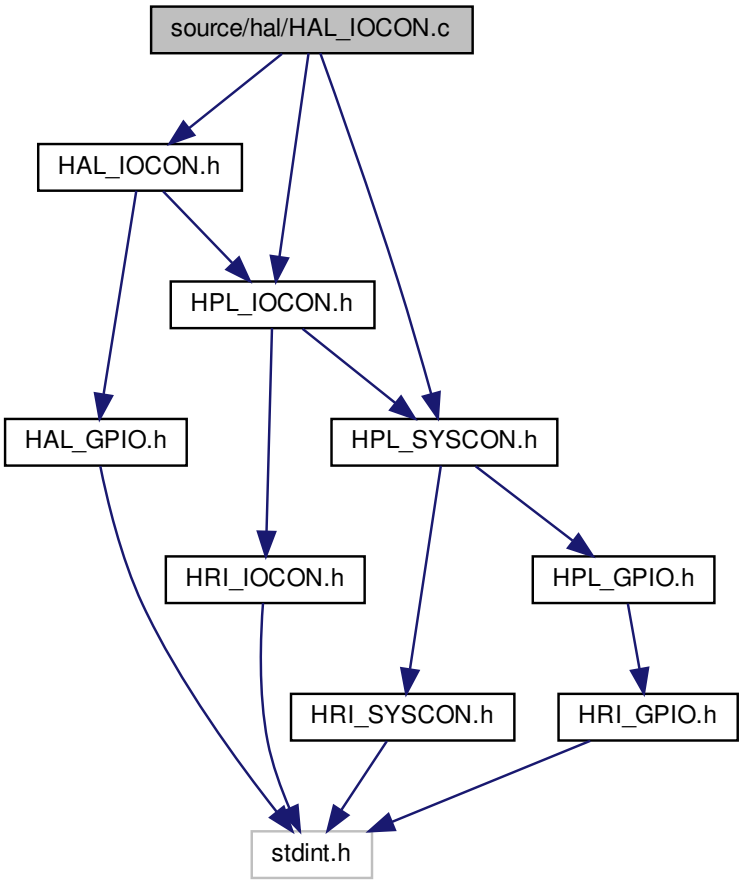
Estado actual de la GPIO

7.44. Referencia del Archivo source/hal/HAL_IOCON.c

Funciones a nivel de aplicacion del periferico IOCON (LPC845)

```
#include <HAL_IOCON.h>
#include <HPL_IOCON.h>
#include <HPL_SYSCON.h>
```

Dependencia gráfica adjunta para HAL_IOCON.c:



Funciones

- void hal_iocon_config_io (hal_gpio_portpin_en portpin, const hal_iocon_config_t *config)
  Configuracion de un pin.

7.44.1. Descripción detallada

Funciones a nivel de aplicacion del periferico IOCON (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.44.2. Documentación de las funciones

7.44.2.1. hal_iocon_config_io()

```
void hal_iocon_config_io (
            hal_gpio_portpin_en portpin,
            const hal_iocon_config_t * config )
```

Configuracion de un pin.

Parámetros

in	*portpin*	Puerto/pin a configurar
in	*pin_config*	Puntero a estructura de configuracion del pin

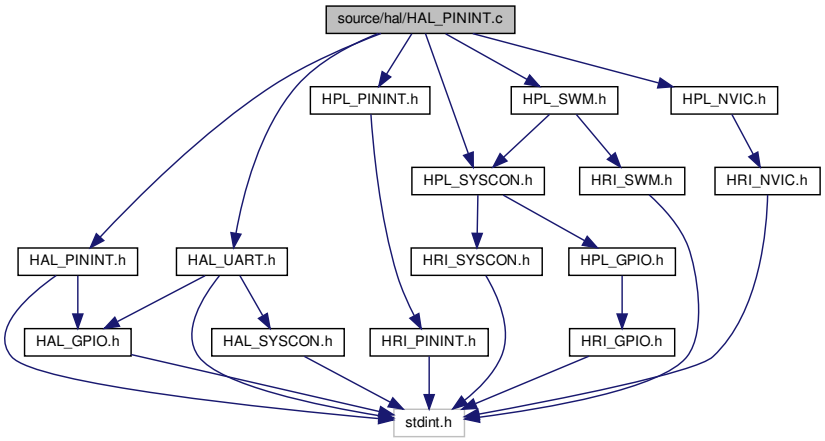
7.45. Referencia del Archivo source/hal/HAL_PININT.c

Funciones a nivel de aplicacion del periferico PININT (LPC845)

```
#include <HAL_PININT.h>
#include <HAL_UART.h>
#include <HPL_PININT.h>
```

```
#include <HPL_SYSCON.h>
#include <HPL_SWM.h>
#include <HPL_NVIC.h>
```

Dependencia gráfica adjunta para HAL_PININT.c:



Funciones

- static void dummy_irq_callback (void)

  Funcion dummy para inicializar los punteros de interrupciones.
- static void hal_pinint_handle_irq (hal_pinint_channel_en channel)

  Manejo de interrupciones para el modulo.
- void hal_pinint_init (void)

  Inicializacion del modulo.
- void hal_pinint_configure_pin_interrupt (const hal_pinint_config_t ∗config)

  Configurar interrupciones de pin.
- void hal_pinint_register_callback (hal_pinint_channel_en channel, void(∗new_callback)(void))

  Registrar callback a llamar en interrupcion de PININTn.
- void PININT0_IRQHandler (void)

  Interrupcion para PININT0.
- void PININT1_IRQHandler (void)

  Interrupcion para PININT1.
- void PININT2_IRQHandler (void)

  Interrupcion para PININT2.
- void PININT3_IRQHandler (void)

  Interrupcion para PININT3.
- void PININT4_IRQHandler (void)

  Interrupcion para PININT4.
- void PININT5_IRQHandler (void)

  Interrupcion para PININT5.
- void PININT6_IRQHandler (void)

  Interrupcion para PININT6 y USART3.
- void PININT7_IRQHandler (void)

  Interrupcion para PININT7 y USART4.

Variables

- static void(∗ **pinint_callbacks** [8])(void)

7.45.1. Descripción detallada

Funciones a nivel de aplicacion del periferico PININT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.45.2. Documentación de las funciones

7.45.2.1. hal_pinint_handle_irq()

```
static void hal_pinint_handle_irq (
            hal_pinint_channel_en channel )  [static]
```

Manejo de interrupciones para el modulo.

Parámetros

in	*channel*	Canal que genero la itnerrupcion

7.45.2.2. hal_pinint_configure_pin_interrupt()

```
void hal_pinint_configure_pin_interrupt (
            const hal_pinint_config_t * config )
```

Configurar interrupciones de pin.

Parámetros

in	*config*	Configuracion de interrupciones de pin

7.45.2.3. hal_pinint_register_callback()

```
void hal_pinint_register_callback (
            hal_pinint_channel_en channel,
            void(*)(void) new_callback )
```

Registrar callback a llamar en interrupcion de PININTn.

Parámetros

in	*channel*	Canal al cual registrar el callback
in	*new_callback*	Puntero a funcion a ejecutar

7.45.3. Documentación de las variables

7.45.3.1. pinint_callbacks

```
void(* pinint_callbacks[8])(void)  [static]
```

Valor inicial:

```
= {
        dummy_irq_callback,
        dummy_irq_callback,
        dummy_irq_callback,
        dummy_irq_callback,
        dummy_irq_callback,
        dummy_irq_callback,
        dummy_irq_callback
}
```

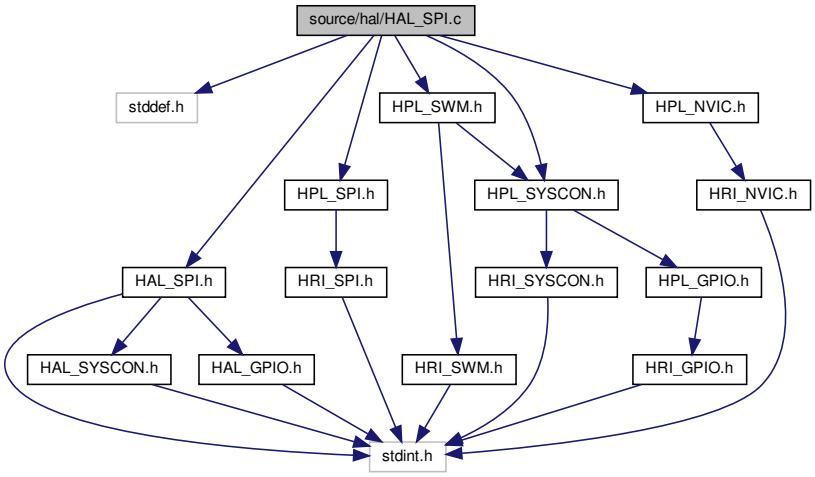
7.46. Referencia del Archivo source/hal/HAL_SPI.c

Funciones a nivel de aplicacion del periferico SPI (LPC845)

```
#include <stddef.h>
#include <HAL_SPI.h>
#include <HPL_SPI.h>
#include <HPL_SWM.h>
#include <HPL_SYSCON.h>
```

```
#include <HPL_NVIC.h>
```

Dependencia gráfica adjunta para HAL_SPI.c:



Funciones

- static void **dummy_irq** (void)
- static void spi_irq_handler (uint8_t inst)

  Manejador generico de interrupciones de SPI.
- void hal_spi_master_mode_init (hal_spi_sel_en inst, const hal_spi_master_mode_config_t ∗config)

  Inicializar SPI en modo master.
- uint16_t hal_spi_master_mode_rx_data (hal_spi_sel_en inst)

  Leer el dato recibido.
- void hal_spi_master_mode_config_tx (hal_spi_sel_en inst, const hal_spi_master_mode_tx_config_t ∗config)

  Configurar la transmision.
- void hal_spi_master_mode_tx_data (hal_spi_sel_en inst, const hal_spi_master_mode_tx_data_t ∗data)

  Transmitir dato.
- void hal_spi_master_mode_register_tx_callback (hal_spi_sel_en inst, void(∗new_callback)(void))

  Actualizar callback en TXRDY.
- void hal_spi_master_mode_register_rx_callback (hal_spi_sel_en inst, void(∗new_callback)(void))

  Actualizar callback en RXRDY.
- void SPI0_IRQHandler (void)

  Manejador de interrupcion de SPI0.
- void SPI1_IRQHandler (void)

  Manejador de interrupcion de SPI1.

Variables

- static void(∗ spi_rx_callback [ ])(void)

  Callbacks registrados a la recepcion de un dato por SPI.
- static void(∗ spi_tx_callback [ ])(void)

  Callbacks registrados a la liberacion del buffer de transmision de SPI.

7.46.1. Descripción detallada

Funciones a nivel de aplicacion del periferico SPI (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.46.2. Documentación de las funciones

7.46.2.1. spi_irq_handler()

```
static void spi_irq_handler (
            uint8_t inst )  [static]
```

Manejador generico de interrupciones de SPI.

Parámetros

in	*inst*	Instancia que genero la interrupcion

7.46.2.2. hal_spi_master_mode_init()

```
void hal_spi_master_mode_init (
            hal_spi_sel_en inst,
            const hal_spi_master_mode_config_t * config )
```

Inicializar SPI en modo master.

Parámetros

in	*inst*	Instancia de SPI a inicializar
in	*config*	Configuracion deseada

7.46.2.3. hal_spi_master_mode_rx_data()

```
uint16_t hal_spi_master_mode_rx_data (
            hal_spi_sel_en inst )
```

Leer el dato recibido.

Parámetros

in	*inst*	Instancia a consultar

Devuelve

Dato recibido

7.46.2.4. hal_spi_master_mode_config_tx()

```
void hal_spi_master_mode_config_tx (
            hal_spi_sel_en inst,
            const hal_spi_master_mode_tx_config_t * config )
```

Configurar la transmision.

Parámetros

in	*inst*	Instancia a configurar
in	*config*	Configuracion para la transmision deseada

7.46.2.5. hal_spi_master_mode_tx_data()

```
void hal_spi_master_mode_tx_data (
            hal_spi_sel_en inst,
            const hal_spi_master_mode_tx_data_t * data )
```

Transmitir dato.

Parámetros

in	*inst*	Instancia a utilizar
in	*data*	Dato a transmitir, con controles asociados

7.46.2.6. hal_spi_master_mode_register_tx_callback()

```
void hal_spi_master_mode_register_tx_callback (
            hal_spi_sel_en inst,
            void(*)(void) new_callback )
```

Actualizar callback en TXRDY.

Parámetros

in	*inst*	Instancia a configurar
in	*new_callback*	Nuevo callback a ejecutar en TXRDY

7.46.2.7. hal_spi_master_mode_register_rx_callback()

```
void hal_spi_master_mode_register_rx_callback (
            hal_spi_sel_en inst,
            void(*)(void) new_callback )
```

Actualizar callback en RXRDY.

Parámetros

in	*inst*	Instancia a configurar
in	*new_callback*	Nuevo callback a ejecutar en RXRDY

7.46.3. Documentación de las variables

7.46.3.1. spi_rx_callback

```
void(* spi_rx_callback[])(void)  [static]
```

Valor inicial:

```
=
{
    dummy_irq,
    dummy_irq
}
```

Callbacks registrados a la recepcion de un dato por SPI.

7.46.3.2. spi_tx_callback

```
void(* spi_tx_callback[])(void)  [static]
```

Valor inicial:

```
=
{
    dummy_irq,
    dummy_irq
}
```

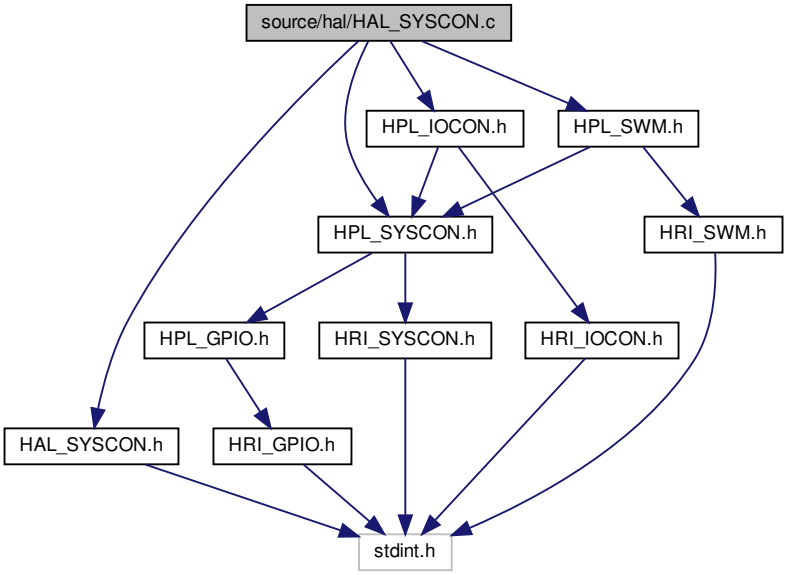
Callbacks registrados a la liberacion del buffer de transmision de SPI.

7.47. Referencia del Archivo source/hal/HAL_SYSCON.c

Funciones a nivel de aplicacion para el SYSCON (LPC845)

```
#include <HAL_SYSCON.h>
#include <HPL_SYSCON.h>
#include <HPL_IOCON.h>
#include <HPL_SWM.h>
```

Dependencia gráfica adjunta para HAL_SYSCON.c:



defines

- #define **XTALIN_PORT** 0
- #define **XTALIN_PIN** 8
- #define **XTALOUT_PORT** 0
- #define **XTALOUT_PIN** 9
- #define **FRO_DIRECT_FREQ** 24e6

Funciones

- uint32_t hal_syscon_get_system_clock (void)

  Obtener la frecuencia actual del main clock.
- uint32_t hal_syscon_get_fro_clock (void)

  Obtener la frecuencia actual del FRO.
- void hal_syscon_config_external_crystal (uint32_t crystal_freq, uint8_t use_as_main)

  Configurar el ext clock a partir de un cristal externo.
- void hal_syscon_config_fro_direct (uint8_t direct, uint8_t use_as_main)

  Configurar el clock FRO.
- void hal_syscon_config_clkout (uint8_t port, uint8_t pin, hal_syscon_clkout_source_sel_en clock_source, uint8_t divider)

  Configurar el pin de clock out (salida de clock hacia afuera)
- void hal_syscon_config_frg (uint8_t inst, hal_syscon_frg_clock_sel_en clock_source, uint32_t mul)

  Configurar el divisor fraccional.
- void hal_syscon_set_peripheral_clock_source (hal_syscon_peripheral_sel_en peripheral, hal_syscon_↩ peripheral_clock_sel_en clock_source)

  Fijar la fuente de clock de un periferico.
- uint32_t hal_syscon_get_peripheral_clock (hal_syscon_peripheral_sel_en peripheral)

  Obtener la frecuencia de clock en Hz configurada para cierto periferico.
- void hal_syscon_set_iocon_glitch_divider (hal_syscon_iocon_glitch_sel_en sel, uint32_t div)

  Configurar divisor para el clock de glitches del IOCON.
- void hal_syscon_config_pll (hal_syscon_pll_source_sel_en clock_source, uint32_t freq)

  Configurar el PLL.
- uint32_t hal_syscon_get_pll_clock (void)

  Obtener frecuencia actual configurada del PLL.

Variables

- static uint32_t current_main_freq = FRO_DIRECT_FREQ / 2

  Frecuencia actual del main clock.
- static uint32_t current_fro_freq = FRO_DIRECT_FREQ / 2

  Frecuencia actual del FRO.
- static uint32_t current_crystal_freq = 0

  Frecuencia del cristal configurada.
- static uint32_t current_frg_freq [2] = { 0, 0 }

  Frecuencia de los FRG.
- static uint32_t current_pll_freq = 0

  Frecuencia del PLL.

7.47.1. Descripción detallada

Funciones a nivel de aplicacion para el SYSCON (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.47.2. Documentación de las funciones

7.47.2.1. hal_syscon_get_system_clock()

```
uint32_t hal_syscon_get_system_clock (
            void  )
```

Obtener la frecuencia actual del main clock.

Devuelve

Frecuencia del main clock en Hz

7.47.2.2. hal_syscon_get_fro_clock()

```
uint32_t hal_syscon_get_fro_clock (
            void  )
```

Obtener la frecuencia actual del FRO.

Devuelve

Frecuencia del FRO en Hz

7.47.2.3. hal_syscon_config_external_crystal()

```
void hal_syscon_config_external_crystal (
            uint32_t crystal_freq,
            uint8_t use_as_main )
```

Configurar el ext clock a partir de un cristal externo.

Parámetros

in	*crystal_freq*	Frecuencia del cristal externo utilizado
in	*use_as_main*	Si es distinto de cero, se utilizara el oscilador a cristal como main clock

7.47.2.4. hal_syscon_config_fro_direct()

```
void hal_syscon_config_fro_direct (
```

```
            uint8_t direct,
            uint8_t use_as_main )
```

Configurar el clock FRO.

Parámetros

in	*direct*	Si es distinto de cero se omite el divisor del FRO
in	*use_as_main*	Si es distinto de cero, se utilizara el FRO como main clock

7.47.2.5. hal_syscon_config_clkout()

```
void hal_syscon_config_clkout (
            uint8_t port,
            uint8_t pin,
            hal_syscon_clkout_source_sel_en clock_source,
            uint8_t divider )
```

Configurar el pin de clock out (salida de clock hacia afuera)

Parámetros

in	*port*	Numero de puerto por donde sacar el clock out
in	*pin*	Numero de pin por donde sacar el clock out
in	*clock_source*	Fuente deseada para la salida clock out
in	*divider*	Divisor deseado para la salida clock out

7.47.2.6. hal_syscon_config_frg()

```
void hal_syscon_config_frg (
            uint8_t inst,
            hal_syscon_frg_clock_sel_en clock_source,
            uint32_t mul )
```

Configurar el divisor fraccional.

El divisor siempre se debe fijar en 256 para estos MCU.

Parámetros

in	*inst*	Instancia de FRG a configurar
in	*clock_source*	Fuente de clock de entrada para el FRG
in	*mul*	Multiplicador deseado

7.47.2.7. hal_syscon_set_peripheral_clock_source()

```
void hal_syscon_set_peripheral_clock_source (
            hal_syscon_peripheral_sel_en peripheral,
            hal_syscon_peripheral_clock_sel_en clock_source )
```

Fijar la fuente de clock de un periferico.

Parámetros

in	*peripheral*	Periferico deseado
in	*clock_source*	Fuente de clock deseada

7.47.2.8. hal_syscon_get_peripheral_clock()

```
uint32_t hal_syscon_get_peripheral_clock (
            hal_syscon_peripheral_sel_en peripheral )
```

Obtener la frecuencia de clock en Hz configurada para cierto periferico.

Parámetros

in	*peripheral*	Periferico deseado

Devuelve

Frecuencia en Hz del clock del periferico

7.47.2.9. hal_syscon_set_iocon_glitch_divider()

```
void hal_syscon_set_iocon_glitch_divider (
            hal_syscon_iocon_glitch_sel_en sel,
            uint32_t div )
```

Configurar divisor para el clock de glitches del IOCON.

Parámetros

in	*sel*	Seleccion de divisor
in	*div*	Valor de division deseado

7.47.2.10. hal_syscon_config_pll()

```
void hal_syscon_config_pll (
            hal_syscon_pll_source_sel_en clock_source,
            uint32_t freq )
```

Configurar el PLL.

Parámetros

in	*clock_source*	Fuente de clock de referencia para el PLL
in	*freq*	Frecuencia deseada de salida del PLL

7.47.2.11. hal_syscon_get_pll_clock()

```
uint32_t hal_syscon_get_pll_clock (
            void  )
```

Obtener frecuencia actual configurada del PLL.

Devuelve

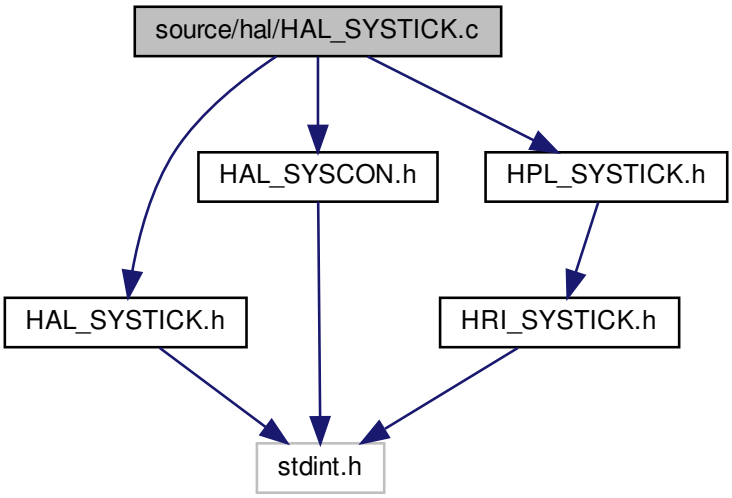
Frecuencia actual del PLL en Hz

7.48. Referencia del Archivo source/hal/HAL_SYSTICK.c

Funciones a nivel de aplicacion para el SYSTICK (LPC845)

```
#include <HAL_SYSTICK.h>
#include <HAL_SYSCON.h>
#include <HPL_SYSTICK.h>
```

Dependencia gráfica adjunta para HAL_SYSTICK.c:



Funciones

- static void dummy_irq (void)
  Dummy function para inicializar los punteros a los callbacks.
- void hal_systick_init (uint32_t tick_us, void(∗callback)(void))
  Inicializacion del SYSTICK.
- void hal_systick_update_callback (void(∗callback)(void))
  Actualizar callback del SYSTICK.
- void SysTick_Handler (void)
  Interrupcion de SYSTICK.

Variables

- static void(∗ systick_callback)(void) = dummy_irq
  Callback a llamar en la interrupcion.

7.48.1. Descripción detallada

Funciones a nivel de aplicacion para el SYSTICK (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.48.2. Documentación de las funciones

7.48.2.1. hal_systick_init()

```
void hal_systick_init (
            uint32_t tick_us,
            void(*)(void) callback )
```

Inicializacion del SYSTICK.

Parámetros

in	*tick_us*	Tiempo en microsegundos deseado para el tick
in	*callback*	Funcion a llamar en cada tick

7.48.2.2. hal_systick_update_callback()

```
void hal_systick_update_callback (
            void(*)(void) callback )
```

Actualizar callback del SYSTICK.

Parámetros

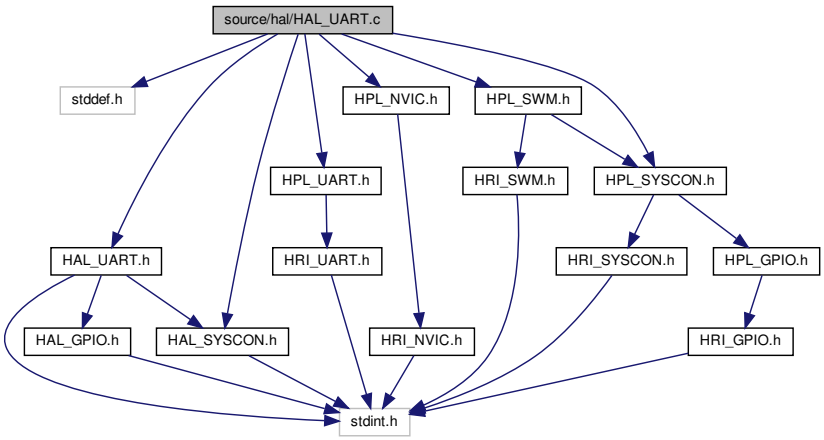
in	*callback*	Nuevo callback a ejecutar en cada tick

7.49. Referencia del Archivo source/hal/HAL_UART.c

Funciones a nivel de aplicacion del periferico UART (LPC845)

```
#include <stddef.h>
#include <HAL_UART.h>
#include <HAL_SYSCON.h>
#include <HPL_UART.h>
#include <HPL_NVIC.h>
#include <HPL_SWM.h>
#include <HPL_SYSCON.h>
```

Dependencia gráfica adjunta para HAL_UART.c:



Funciones

- static void dummy_callback (void)

  Llamado a funcion dummy para irq iniciales.
- static uint16_t hal_uart_calculate_brgval (uint32_t uart_clock, uint32_t baudrate, uint8_t oversampling)

  Calculo del valor para el registro de Baudrate.
- static void **hal_uart_handle_irq** (uint8_t inst)
- void hal_uart_init (uint8_t inst, const hal_uart_config_t ∗config)

  Inicializar UART con los parametros deseados.
- hal_uart_tx_result hal_uart_tx_byte (uint8_t inst, uint32_t data)

  Transmitir un dato mediante la UART.
- hal_uart_rx_result hal_uart_rx_byte (uint8_t inst, uint32_t ∗data)

  Recibir un dato de la UART.
- void hal_uart_register_rx_callback (uint8_t inst, void(∗new_callback)(void))

  Registrar el callback a ser llamado en la recepcion de un dato por UART.
- void hal_uart_register_tx_callback (uint8_t inst, void(∗new_callback)(void))

  Registrar el callback a ser llamado una vez finalizada la transmision de un dato por UART.
- void UART0_IRQHandler (void)

  Interrupcion de UART0.
- void UART1_IRQHandler (void)

  Interrupcion de UART1.
- void UART2_IRQHandler (void)

  Interrupcion de UART2.
- void UART3_irq (void)

  Interrupcion de UART3.
- void UART4_irq (void)

  Interrupcion de UART4.

Variables

- static void(∗ uart_rx_callback [ ])(void)

  Callbacks registrados a la recepcion de un dato por UART.
- static void(∗ uart_tx_callback [ ])(void)

  Callbacks registrados a la finalizacion de transmision de un dato por UART.

7.49.1. Descripción detallada

Funciones a nivel de aplicacion del periferico UART (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.49.2. Documentación de las funciones

7.49.2.1. hal_uart_calculate_brgval()

```
static uint16_t hal_uart_calculate_brgval (
            uint32_t uart_clock,
            uint32_t baudrate,
            uint8_t oversampling )  [inline], [static]
```

Calculo del valor para el registro de Baudrate.

Parámetros

in	*uart_clock*	Clock asociado con la UART.
in	*baudrate*	Baudrate deseado a calcular.
in	*oversampling*	Oversampling deseado para la UART.

Devuelve

Valor a poner en el registro BRG.

7.49.2.2. hal_uart_init()

```
void hal_uart_init (
            uint8_t inst,
            const hal_uart_config_t * config )
```

Inicializar UART con los parametros deseados.

Parámetros

in	*inst*	Que instancia de UART inicializar
in	*config*	Puntero a configuracion de la UART

7.49.2.3. hal_uart_tx_byte()

```
hal_uart_tx_result hal_uart_tx_byte (
            uint8_t inst,
            uint32_t data )
```

Transmitir un dato mediante la UART.

Parámetros

in	*inst*	Que instancia de UART usar
in	*data*	Dato a transmitir. Puede ser de 7, 8 o 9 bits

7.49.2.4. hal_uart_rx_byte()

```
hal_uart_rx_result hal_uart_rx_byte (
            uint8_t inst,
            uint32_t * data )
```

Recibir un dato de la UART.

Parámetros

in	*inst*	Que instancia de UART usar
in	*data*	Puntero a donde guardar el dato recibido

Devuelve

Estado de la recepcion

7.49.2.5. hal_uart_register_rx_callback()

```
void hal_uart_register_rx_callback (
            uint8_t inst,
            void(*)(void) new_callback )
```

Registrar el callback a ser llamado en la recepcion de un dato por UART.

Parámetros

in	*inst*	A que instancia de UART registrar el callback
in	*new_callback*	Puntero a funcion a llamar cada vez que se recibe un dato por UART

7.49.2.6. hal_uart_register_tx_callback()

```
void hal_uart_register_tx_callback (
            uint8_t inst,
            void(*)(void) new_callback )
```

Registrar el callback a ser llamado una vez finalizada la transmision de un dato por UART.

Parámetros

in	*inst*	A que instancia de UART registrar el callback
in	*new_callback*	Puntero a funcion a llamar cada vez que se termina de enviar un dato por UART

7.49.3. Documentación de las variables

7.49.3.1. uart_rx_callback

```
void(* uart_rx_callback[])(void)  [static]
```

Valor inicial:

```
=
{
        dummy_callback,
        dummy_callback,
        dummy_callback,
        dummy_callback,
        dummy_callback
}
```

Callbacks registrados a la recepcion de un dato por UART.

7.49.3.2. uart_tx_callback

```
void(* uart_tx_callback[])(void)  [static]
```

Valor inicial:

```
=
{
        dummy_callback,
        dummy_callback,
        dummy_callback,
        dummy_callback,
        dummy_callback
}
```

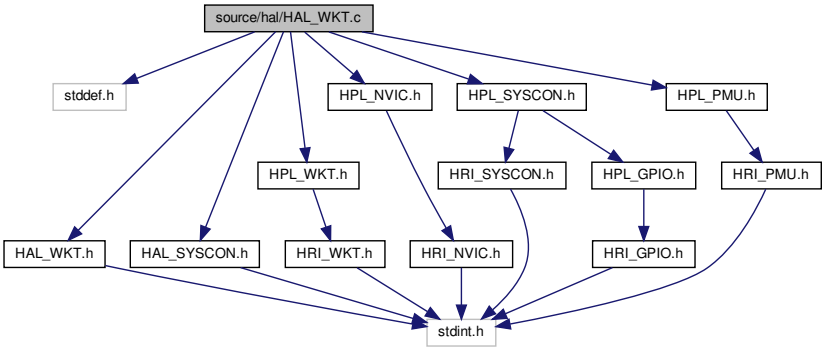
Callbacks registrados a la finalizacion de transmision de un dato por UART.

7.50. Referencia del Archivo source/hal/HAL_WKT.c

Funciones a nivel de aplicacion del periferico WKT (LPC845)

```
#include <stddef.h>
#include <HAL_WKT.h>
#include <HAL_SYSCON.h>
#include <HPL_WKT.h>
#include <HPL_NVIC.h>
#include <HPL_SYSCON.h>
#include <HPL_PMU.h>
```

Dependencia gráfica adjunta para HAL_WKT.c:



defines

- #define **HAL_WKT_DIVIDE_VALUE** (16)
- #define **HAL_WKT_LOW_POWER_OSC_FREQ** (10e3)

Funciones

- static void **dummy_irq** (void)
- void hal_wkt_init (hal_wkt_clock_source_en clock_sel, uint32_t ext_clock_value, void(∗callback)(void))

  Inicializar el WKT.
- void **hal_wkt_select_clock_source** (hal_wkt_clock_source_en clock_sel, uint32_t ext_clock_value)
- void hal_wkt_register_callback (void(∗new_callback)(void))

  Registrar un callback para la interrupcion del WKT.
- void **hal_wkt_start_count** (uint32_t time_useg)
- void **hal_wkt_start_count_with_value** (uint32_t value)
- void WKT_IRQHandler (void)

  Interrupcion de WKT.

Variables

- static hal_wkt_clock_source_en **current_clock_source** = HAL_WKT_CLOCK_SOURCE_FRO_DIV
- static uint32_t **current_ext_clock** = 0
- static void(∗ **hal_wkt_irq_callback**)(void) = dummy_irq

7.50.1. Descripción detallada

Funciones a nivel de aplicacion del periferico WKT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.50.2. Documentación de las funciones

7.50.2.1. hal_wkt_init()

```
void hal_wkt_init (
            hal_wkt_clock_source_en clock_sel,
            uint32_t ext_clock_value,
            void(*)(void) callback )
```

Inicializar el WKT.

Parámetros

in	*clock_sel*	Seleccion de clock deseada para el WKT
in	*ext_clock_value*	Valor de clock externo (si la seleccion es interna, no importa este parametro)
in	*callback*	Callback a ejecutar en la interrupcion del WKT

7.50.2.2. hal_wkt_register_callback()

```
void hal_wkt_register_callback (
            void(*)(void) new_callback )
```

Registrar un callback para la interrupcion del WKT.

Parámetros

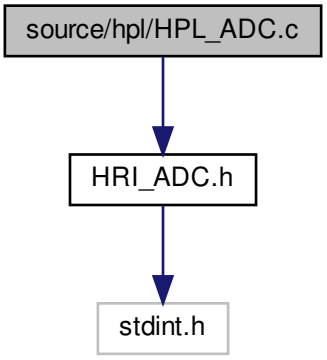
in	*new_callback*	Nuevo callback para la interrupcion del WKT

7.51. Referencia del Archivo source/hpl/HPL_ADC.c

Funciones a nivel de abstraccion de periferico para el ADC (LPC845)

```
#include <HRI_ADC.h>
```

Dependencia gráfica adjunta para HPL_ADC.c:



Variables

- volatile ADC_per_t *const ADC = (ADC_per_t *) ADC_BASE

  Periferico ADC.

7.51.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el ADC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

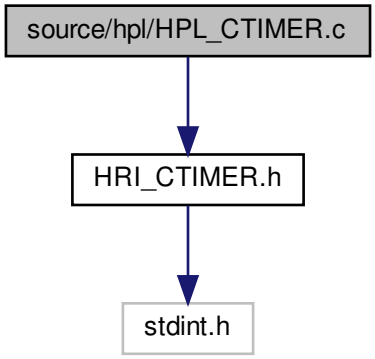
1.0

7.52. Referencia del Archivo source/hpl/HPL_CTIMER.c

Funciones a nivel de abstraccion del periferico CTIMER (LPC845)

```
#include <HRI_CTIMER.h>
```

Dependencia gráfica adjunta para HPL_CTIMER.c:



Variables

- volatile CTIMER_per_t ∗const CTIMER = (volatile CTIMER_per_t ∗) CTIMER_BASE

  Periferico CTIMER.

7.52.1. Descripción detallada

Funciones a nivel de abstraccion del periferico CTIMER (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

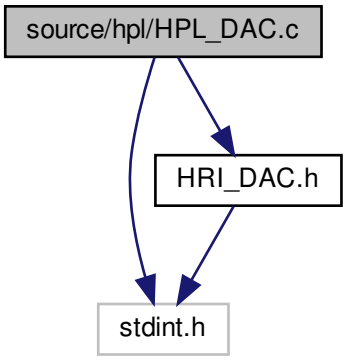
1.0

7.53. Referencia del Archivo source/hpl/HPL_DAC.c

Funciones a nivel de abstraccion de periferico para el DAC (LPC845)

```
#include <stdint.h>
#include <HRI_DAC.h>
```

Dependencia gráfica adjunta para HPL_DAC.c:



Variables

- volatile DAC_per_t *const DAC [ ]
  Perifericos DAC.

7.53.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el DAC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.53.2. Documentación de las variables

7.53.2.1. DAC

```
volatile DAC_per_t* const DAC[]
```

Valor inicial:

```
= {
        (DAC_per_t *) DAC0_BASE,
        (DAC_per_t *) DAC1_BASE
}
```

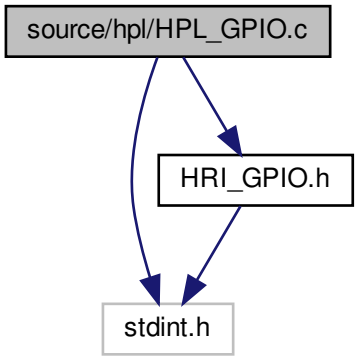
Perifericos DAC.

7.54. Referencia del Archivo source/hpl/HPL_GPIO.c

Funciones a nivel de abstraccion de periferico para el GPIO (LPC845)

```
#include <stdint.h>
#include <HRI_GPIO.h>
```

Dependencia gráfica adjunta para HPL_GPIO.c:



Variables

- volatile GPIO_per_t *const GPIO = (GPIO_per_t *) GPIO_BASE

  Periferico GPIO.

7.54.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el GPIO (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

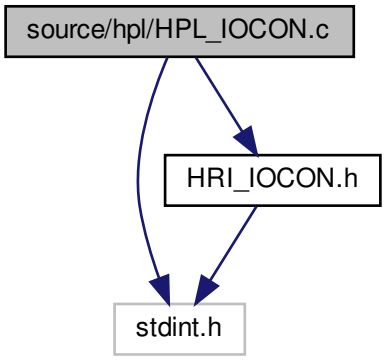
1.0

7.55. Referencia del Archivo source/hpl/HPL_IOCON.c

Funciones a nivel de abstraccion de periferico para el IOCON (LPC845)

```
#include <stdint.h>
#include <HRI_IOCON.h>
```

Dependencia gráfica adjunta para HPL_IOCON.c:



Variables

- volatile IOCON_per_t ∗const IOCON = (IOCON_per_t ∗) IOCON_BASE

  Periferico IOCON.
- volatile IOCON_PIO_reg_t dummy_reg

  Registro dummy para los pines no disponibles en el encapsulado.
- volatile IOCON_PIO_reg_t ∗const IOCON_PIN_TABLE [2][32]

  Tabla de registros de configuracion.

7.55.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el IOCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

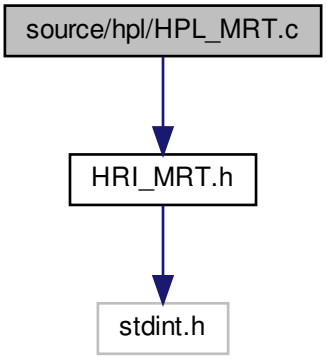
Versión

1.0

7.56. Referencia del Archivo source/hpl/HPL_MRT.c

Funciones a nivel de abstraccion de periferico para el MRT (LPC845)

```
#include <HRI_MRT.h>
```
Dependencia gráfica adjunta para HPL_MRT.c:



Variables

- volatile MRT_per_t ∗const MRT = (MRT_per_t ∗) MRT_BASE

  Periferico MRT.

7.56.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el MRT (LPC845)

Autor

Augusto Santini

Fecha

4/2020

Versión

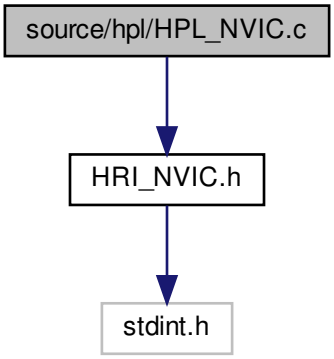
1.0

7.57. Referencia del Archivo source/hpl/HPL_NVIC.c

Funciones a nivel de abstraccion de periferico para el NVIC (LPC845)

```
#include <HRI_NVIC.h>
```

Dependencia gráfica adjunta para HPL_NVIC.c:



Variables

- volatile NVIC_per_t *const NVIC = (NVIC_per_t *) NVIC_BASE

  Periferico NVIC.

7.57.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el NVIC (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

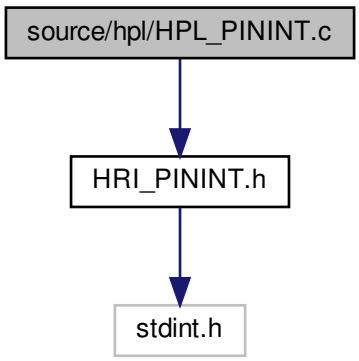
1.0

7.58. Referencia del Archivo source/hpl/HPL_PININT.c

Funciones a nivel de abstraccion de periferico para el PININT (LPC845)

```
#include <HRI_PININT.h>
```

Dependencia gráfica adjunta para HPL_PININT.c:



Variables

- volatile PININT_per_t ∗const PININT = (PININT_per_t ∗) PININT_BASE

  Periferico PININT.

7.58.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el PININT (LPC845)

Autor

Augusto Santini

Fecha

6/2019

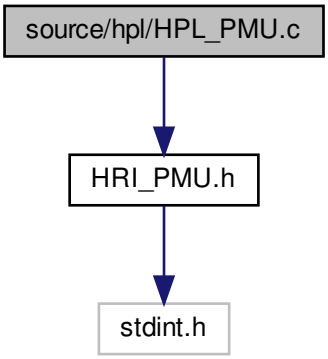
Versión

1.0

7.59. Referencia del Archivo source/hpl/HPL_PMU.c

Funciones a nivel de abstraccion de periferico para el PMU (LPC845)

```
#include <HRI_PMU.h>
```
Dependencia gráfica adjunta para HPL_PMU.c:



Variables

- volatile SCR_reg_t ∗const SCR = (volatile SCR_reg_t ∗) SCR_REG_BASE

  Registro SCR.
- volatile PMU_per_t ∗const PMU = (volatile PMU_per_t ∗) PMU_BASE

  Periferico PMU.

7.59.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el PMU (LPC845)

Autor

Augusto Santini

Fecha

4/2020

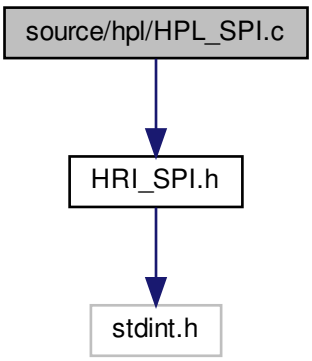
Versión

1.0

7.60. Referencia del Archivo source/hpl/HPL_SPI.c

Funciones a nivel de abstraccion de periferico para el SPI (LPC845)

```
#include <HRI_SPI.h>
```
Dependencia gráfica adjunta para HPL_SPI.c:



Variables

- volatile SPI_per_t ∗const SPI [ ]

  Perifericos SPI.

7.60.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el SPI (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

7.60.2. Documentación de las variables

7.60.2.1. SPI

```
volatile SPI_per_t* const SPI[]
```

Valor inicial:

```
= {
        (SPI_per_t *) SPI0_BASE,
        (SPI_per_t *) SPI1_BASE
}
```

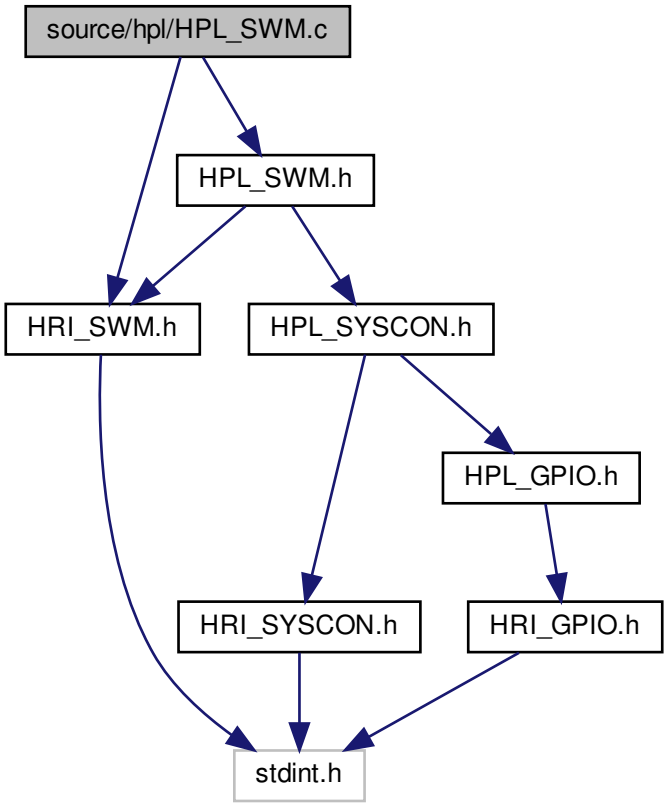
Perifericos SPI.

7.61. Referencia del Archivo source/hpl/HPL_SWM.c

Funciones a nivel de abstraccion de periferico para el SWM (LPC845)

```
#include <HPL_SWM.h>
#include <HRI_SWM.h>
```

Dependencia gráfica adjunta para HPL_SWM.c:



Variables

- volatile SWM_per_t ∗const SWM = (SWM_per_t ∗) SWM_BASE
  Periferico SWM.

7.61.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el SWM (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

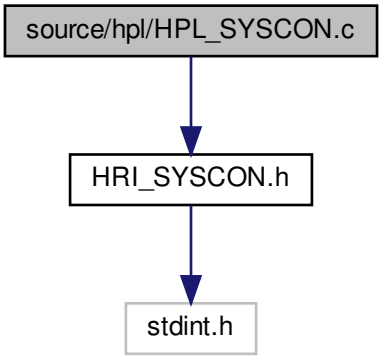
1.0

7.62. Referencia del Archivo source/hpl/HPL_SYSCON.c

Funciones a nivel de abstraccion de periferico para el SYSCON (LPC845)

```
#include <HRI_SYSCON.h>
```

Dependencia gráfica adjunta para HPL_SYSCON.c:



Variables

- volatile SYSCON_per_t ∗const SYSCON = (SYSCON_per_t ∗) SYSCON_BASE

  Periferico SYSCON.

7.62.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el SYSCON (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

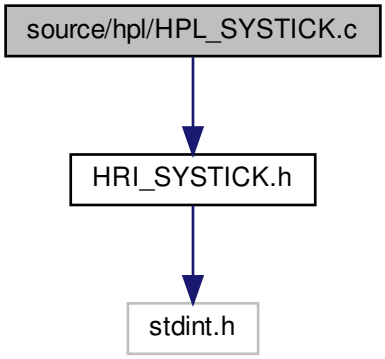
1.0

7.63. Referencia del Archivo source/hpl/HPL_SYSTICK.c

Funciones a nivel de abstraccion de periferico para el SYSTICK (LPC845)

```
#include <HRI_SYSTICK.h>
```

Dependencia gráfica adjunta para HPL_SYSTICK.c:



Variables

- volatile SYSTICK_reg_t ∗const SYSTICK = (SYSTICK_reg_t ∗) SYSTICK_BASE

  Periferico SYSTICK.

7.63.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el SYSTICK (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

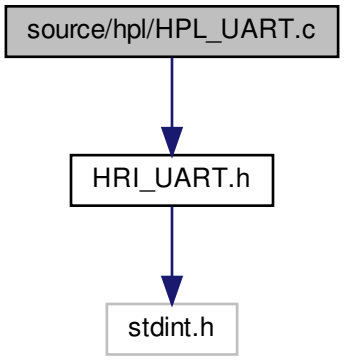
1.0

7.64. Referencia del Archivo source/hpl/HPL_UART.c

Funciones a nivel de abstraccion de periferico para el UART (LPC845)

```
#include <HRI_UART.h>
```

Dependencia gráfica adjunta para HPL_UART.c:



Variables

- volatile UART_per_t ∗const UART [ ]
  Perifericos USART.

7.64.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el UART (LPC845)

Autor

Augusto Santini

Fecha

6/2019

Versión

1.0

7.64.2. Documentación de las variables

7.64.2.1. UART

```
volatile UART_per_t* const UART[]
```

Valor inicial:

```
= {
        (UART_per_t *) UART0_BASE,
        (UART_per_t *) UART1_BASE,
        (UART_per_t *) UART2_BASE,
        (UART_per_t *) UART3_BASE,
        (UART_per_t *) UART4_BASE
}
```

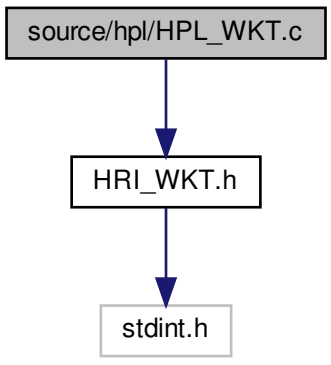
Perifericos USART.

7.65. Referencia del Archivo source/hpl/HPL_WKT.c

Funciones a nivel de abstraccion de periferico para el WKT (LPC845)

```
#include <HRI_WKT.h>
```

Dependencia gráfica adjunta para HPL_WKT.c:



Variables

- volatile WKT_per_t *const WKT = (volatile WKT_per_t *) WKT_BASE

  Periferico WKT.

7.65.1. Descripción detallada

Funciones a nivel de abstraccion de periferico para el WKT (LPC845)

Autor

Augusto Santini

Fecha

3/2020

Versión

1.0

Capítulo 8

Documentación de ejemplos

8.1. Ejemplo_ADC.c

Ejemplo sobre la utilización del *ADC.El* programa utiliza el clock por default con el que comienza el microcontrolador, es decir, el *Free Running Oscillator* funcionando a $12MHz$.

El periférico será configurado con las siguientes características:

- Funcionamiento **sincrónico**
- Frecuencia de muestreo de $1Mhz$
- Modo bajo consumo inhabilitado

La secuencia A es configurada para generar conversiones en los canales 0 y 8:

- El canal 0 está conectado al preset propio del stick de desarrollo (Puerto 0 pin 7)
- El canal 8 está ubicado en el pin número 3 (Puerto 0 pin 18) y se le puede conectar un preset externo entre VDD y GND.

Además, la secuencia tendrá la siguiente configuración:

- Trigger: Únicamente se disparan conversiones por software
- Bypass sincronismo: Sí
- Modo de interrupción: Cuando termina la secuencia completa
- Burst: Inhabilitado
- Un trigger dispara: Una conversión de secuencia completa
- Secuencia A como baja prioridad: No

Una vez inicializado el periférico, se configura el periférico *Systick* para interrumpir cada $1mseg$ y mediante su manejador se lleva la cuenta de los milisegundos transcurridos. Una vez transcurridos $1000mseg$, se dispara una conversión de *ADC*, y sus resultados se guardan en dos variables globales.

Ubicando un *breakpoint* adecuadamente, se pueden leer los resultados de las conversiones ya ubicadas en las variables globales.

```
#include <cr_section_macros.h>
#include <stddef.h>
#include <HAL_ADC.h>
#include <HAL_SYSTICK.h>

/* Máscara de configuración de canales habilitados para la secuencia a configurar */
#define     ADC_CHANNELS                ((1 « 0) | (1 « 8))

/* Macro para definir el tiempo de interrupción del \e Systick en \b microsegundos */
#define     TICK_TIME_USEG              (1000)

/* Frecuencia de muestreo a utilizar por el ADC */
#define     ADC_SAMPLE_FREQ             (1000000)

#define     ADC_SEQUENCE                (HAL_ADC_SEQUENCE_SEL_A)

/* Tiempo de disparo de conversiones de \e ADC en \b milisegundos */
#define     ADC_CONVERSION_TIME_MSEG    (1000)

static void adc_callback(void);

static void systick_callback(void);

static uint8_t flag_secuencia_adc_completada = 0; /* Flag para indicar finalización de secuencia de
       conversión de \e ADC */

/*
 * Variables para guardar los resultados de la secuencia de conversión
 */
static hal_adc_sequence_result_t resultados_conversion_adc[2];

/* Configuración de la secuencia. Como no va a cambiar es declarada \e const */
static const hal_adc_sequence_config_t adc_config =
{
    .channels = ADC_CHANNELS,
    .trigger = HAL_ADC_TRIGGER_SEL_NONE,
    .trigger_pol = HAL_ADC_TRIGGER_POL_SEL_NEGATIVE_EDGE,
    .sync_bypass = HAL_ADC_SYNC_SEL_BYPASS_SYNC,
    .mode = HAL_ADC_INTERRUPT_MODE_EOS,
    .burst = 0,
    .single_step = 0,
    .low_priority = 0,
    .callback = adc_callback
};

/*
 * @brief Punto de entrada del programa
 * @return Nunca deberia terminar esta función
 */
int main(void)
{
    // Inicialización del periférico en modo SINCRÓNICO
    hal_adc_init_sync_mode(ADC_SAMPLE_FREQ,
      HAL_ADC_LOW_POWER_MODE_DISABLED);

    // Configuración de la secuencia a utilizar
    hal_adc_config_sequence(ADC_SEQUENCE, &adc_config);

    // Inicialización del \e Systick con el tiempo de tick adecuado
    hal_systick_init(TICK_TIME_USEG, systick_callback);

    while(1)
    {
        if(flag_secuencia_adc_completada == 1) // Esto o hacer "if(flag_secuencia_adc_completada)" es
       indistinto
        {
            uint32_t variable_auxiliar;

            flag_secuencia_adc_completada = 0;

            (void) variable_auxiliar; // Esta línea es ideal para colocar el breakpoint!
        }
    }

    return 0;
}

/*
 * @brief Callback a ejecutar en cada tick del \e Systick
 */
static void systick_callback(void)
{
    static uint32_t contador_disparo_adc = 0;

    // Conteo con valor límite
    contador_disparo_adc = (contador_disparo_adc + 1) % ADC_CONVERSION_TIME_MSEG;
```

```
    if(contador_disparo_adc == 0) // Esto o hacer "if(!contador_disparo_adc)" es indistinto
    {
        hal_adc_start_sequence(ADC_SEQUENCE);
    }
}

/*
 * @brief Callback a ejecutar en cada finalización de conversión de \b secuencia de \e ADC
 */
static void adc_callback(void)
{
    // Obtención de resultados de conversión
    hal_adc_get_sequence_result(ADC_SEQUENCE, resultados_conversion_adc);

    flag_secuencia_adc_completada = 1;
}
```

Índice alfabético