HARMcksL: ARM HAL toolbox (yet STM32 oriented)

1.2

Generated by Doxygen 1.8.13

Contents

1 Weak Functions List **2**

2 Class Index **2**

2.1 Class List 2

3 File Index **2**

3.1 File List 2

4 Class Documentation **3**

4.1 DateTime Struct Reference 3

4.1.1 Detailed Description 3

4.1.2 Member Data Documentation 4

4.2 GPIO_in Struct Reference 5

4.2.1 Detailed Description 5

4.2.2 Member Data Documentation 5

4.3 logicPWM Struct Reference 7

4.3.1 Detailed Description 7

4.3.2 Member Data Documentation 8

5 File Documentation **9**

5.1 exceptions.c File Reference 9

5.1.1 Detailed Description 10

5.1.2 Function Documentation 10

5.2 exceptions.h File Reference 12

5.2.1 Detailed Description 13

5.2.2 Macro Definition Documentation 14

5.2.3 Function Documentation 14

5.3 GPIO_ex.c File Reference 17

5.3.1 Detailed Description 17

5.3.2 Macro Definition Documentation 18

5.3.3 Function Documentation 18

5.4 GPIO_ex.h File Reference 19
5.4.1 Detailed Description 21
5.4.2 Typedef Documentation 21
5.4.3 Enumeration Type Documentation 21
5.4.4 Function Documentation 22
5.5 PWM.c File Reference 26
5.5.1 Detailed Description 27
5.5.2 Function Documentation 27
5.6 PWM.h File Reference 34
5.6.1 Detailed Description 36
5.6.2 Typedef Documentation 36
5.6.3 Function Documentation 36
5.7 RTC_ex.c File Reference 47
5.7.1 Detailed Description 48
5.7.2 Function Documentation 49
5.8 RTC_ex.h File Reference 49
5.8.1 Detailed Description 50
5.8.2 Function Documentation 51
5.9 stdream_rdir.c File Reference 51
5.9.1 Detailed Description 52
5.9.2 Function Documentation 53
5.9.3 Variable Documentation 54
5.10 stdream_rdir.h File Reference 55
5.10.1 Detailed Description 56
5.10.2 Macro Definition Documentation 56
5.10.3 Function Documentation 57
5.10.4 Variable Documentation 59
5.11 time_utils.c File Reference 59
5.11.1 Detailed Description 60
5.11.2 Function Documentation 60
5.12 time_utils.h File Reference 62
5.12.1 Detailed Description 63
5.12.2 Typedef Documentation 64
5.12.3 Function Documentation 64
5.13 UART_term.c File Reference 66
5.13.1 Detailed Description 67
5.13.2 Function Documentation 67
5.13.3 Variable Documentation 69
5.14 UART_term.h File Reference 70
5.14.1 Detailed Description 71
5.14.2 Macro Definition Documentation 72
5.14.3 Function Documentation 72
5.14.4 Variable Documentation 77

Index 79

1 Weak Functions List

Member SERIAL_DBG_Message_Handler (const char ∗msg, uint8_t len)
This function is implemented as weak to be implemented in projects (weak one only prints & flushes the buffer)

2 Class Index

2.1 Class List

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

DateTime
Basic Date & Time struct 3
GPIO_in
GPIO input structure 5
logicPWM
Software PWM on GPIO struct 7

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

exceptions.c
Debug tool helpers functions 9
exceptions.h
Debug tool and helpers declaration 12
GPIO_ex.c
Simple extension for GPIOs 17
GPIO_ex.h
Simple extension for GPIOs 19
PWM.c
Straightforward PWM handling 26
PWM.h
Straightforward PWM handling 34
RTC_ex.c
Basic RTC handling 47

RTC_ex.h
Basic RTC handling 49

stdream_rdir.c
Stream redirection 51

stdream_rdir.h
Stream redirection header 55

time_utils.c
Time related utilities 59

time_utils.h
Time related utilities 62

UART_term.c
UART terminal 66

UART_term.h
UART terminal header 70

4 Class Documentation

4.1 DateTime Struct Reference

Basic Date & Time struct.

```
#include <time_utils.h>
```

Public Attributes

- uint16_t Year

  Year.
- uint8_t Month

  Month.
- uint8_t Day

  Day.
- uint8_t Weekday

  Weekday.
- uint8_t Hours

  Hours.
- uint8_t Minutes

  Minutes.
- uint8_t Seconds

  Seconds.

4.1.1 Detailed Description

Basic Date & Time struct.

4.1.2 Member Data Documentation

4.1.2.1 Day

`uint8_t DateTime::Day`

Day.

4.1.2.2 Hours

`uint8_t DateTime::Hours`

Hours.

4.1.2.3 Minutes

`uint8_t DateTime::Minutes`

Minutes.

4.1.2.4 Month

`uint8_t DateTime::Month`

Month.

4.1.2.5 Seconds

`uint8_t DateTime::Seconds`

Seconds.

4.1.2.6 Weekday

`uint8_t DateTime::Weekday`

Weekday.

4.1.2.7 Year

```
uint16_t DateTime::Year
```

Year.

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

- time_utils.h

4.2 GPIO_in Struct Reference

GPIO input structure.

```
#include <GPIO_ex.h>
```

Public Attributes

- bool in

  Input value.
- eEdge edge

  Input edge.
- bool mem

  Memo value.
- bool done

  State change done.
- uint32_t hIn

  Filter time.
- struct {
    GPIO_TypeDef ∗ GPIOx
      HAL GPIO instance.
    uint16_t GPIO_Pin
      HAL GPIO pin.
    uint16_t filt
      Filter time (ms)
  } cfg

4.2.1 Detailed Description

GPIO input structure.

4.2.2 Member Data Documentation

4.2.2.1 cfg

```
struct { ...  } GPIO_in::cfg
```

4.2.2.2 done

```
bool GPIO_in::done
```

State change done.

4.2.2.3 edge

```
eEdge GPIO_in::edge
```

Input edge.

4.2.2.4 filt

```
uint16_t GPIO_in::filt
```

Filter time (ms)

4.2.2.5 GPIO_Pin

```
uint16_t GPIO_in::GPIO_Pin
```

HAL GPIO pin.

4.2.2.6 GPIOx

```
GPIO_TypeDef* GPIO_in::GPIOx
```

HAL GPIO instance.

4.2.2.7 hIn

```
uint32_t GPIO_in::hIn
```

Filter time.

4.2.2.8 in

```
bool GPIO_in::in
```

Input value.

4.2.2.9 mem

```
bool GPIO_in::mem
```

Memo value.

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

- GPIO_ex.h

4.3 logicPWM Struct Reference

Software PWM on GPIO struct.

```
#include <PWM.h>
```

Public Attributes

- uint16_t cntr

    Counter.
- uint16_t duty

    Current Duty cycle.
- struct {
    TIM_HandleTypeDef ∗ pTim
      Timer instance (for reference)
    GPIO_TypeDef ∗ GPIOx
      Port of emulated PWM pin.
    uint16_t GPIO_Pin
      Pin mask on port.
    uint16_t tim_freq
      Timer frequency (for reference)
    uint16_t duty
      Duty Cycle (effective when new period starts)
    uint16_t per
      Overflow threshold (emulated PWM period)
    bool polarity
      Output polarity.
  } cfg

4.3.1 Detailed Description

Software PWM on GPIO struct.

4.3.2 Member Data Documentation

4.3.2.1 cfg

```
struct { ...  } logicPWM::cfg
```

4.3.2.2 cntr

```
uint16_t logicPWM::cntr
```

Counter.

4.3.2.3 duty

```
uint16_t logicPWM::duty
```

Current Duty cycle.

Duty Cycle (effective when new period starts)

4.3.2.4 GPIO_Pin

```
uint16_t logicPWM::GPIO_Pin
```

Pin mask on port.

4.3.2.5 GPIOx

```
GPIO_TypeDef* logicPWM::GPIOx
```

Port of emulated PWM pin.

4.3.2.6 per

```
uint16_t logicPWM::per
```

Overflow threshold (emulated PWM period)

4.3.2.7 polarity

```
bool logicPWM::polarity
```

Output polarity.

4.3.2.8 pTim

```
TIM_HandleTypeDef* logicPWM::pTim
```

Timer instance (for reference)

4.3.2.9 tim_freq

```
uint16_t logicPWM::tim_freq
```

Timer frequency (for reference)

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

- PWM.h

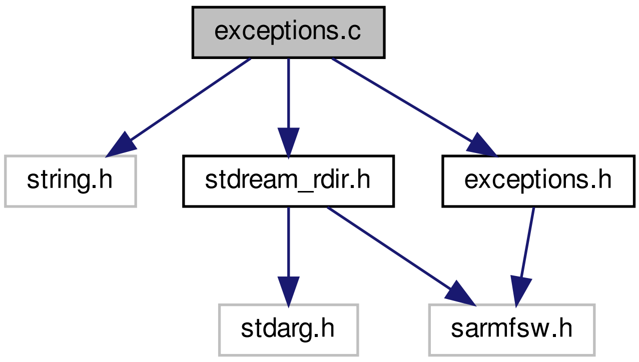
5 File Documentation

5.1 exceptions.c File Reference

Debug tool helpers functions.

```
#include <string.h>
#include "stdream_rdir.h"
#include "exceptions.h"
```

Include dependency graph for exceptions.c:



Functions

- void stackDump (uint32_t stack[ ])

  prints contents of stack
- void HardFault_Handler_callback (uint32_t stack[ ])

  prints informations about current Hard Fault exception
- void Error_Handler_callback (uint32_t stack[ ])

  prints informations about current Hard Fault exception

5.1.1 Detailed Description

Debug tool helpers functions.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.1.2 Function Documentation

5.1.2.1 Error_Handler_callback()

```
void Error_Handler_callback (
            uint32_t stack[ ] )
```

prints informations about current Hard Fault exception

Parameters

in	*stack*	- pointer to stack address

Note

HardFault_Handler_callback should not be called directly use exception_Handler() which prepares pointer to current stack instead

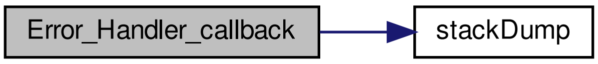
Warning

Depending how arm is fucked up, informations may not be printed, at least, you could inspect exception and stack through debug breakpoint

Returns

Never (anyways, arm fubared!)

Here is the call graph for this function:



5.1.2.2 HardFault_Handler_callback()

```
void HardFault_Handler_callback (
             uint32_t stack[ ] )
```

prints informations about current Hard Fault exception

Parameters

in	*stack*	- pointer to stack address

Note

HardFault_Handler_callback should not be called directly use exception_Handler() which prepares pointer to current stack instead

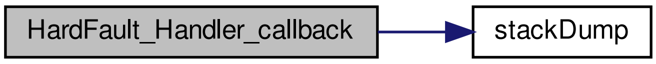
Warning

Depending how arm is fucked up, informations may not be printed, at least, you could inspect exception and stack through debug breakpoint

Returns

Never (anyways, arm fubared!)

Here is the call graph for this function:



5.1.2.3 stackDump()

```
void stackDump (
             uint32_t stack[] )
```

prints contents of stack

Parameters

in	*stack*	- pointer to stack address

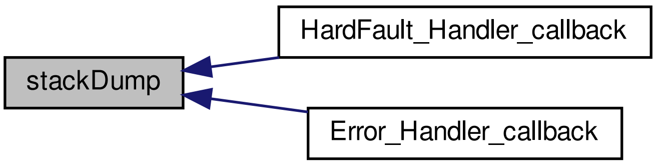
Note

stackDump should not be called directly, unless a particular stack is needed use dump_stack() which prepares pointer to current stack instead

Returns

Nothing

Here is the caller graph for this function:

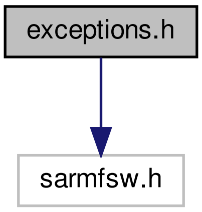


5.2 exceptions.h File Reference

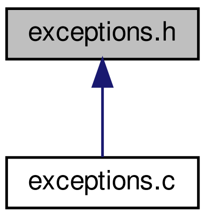
Debug tool and helpers declaration.

```
#include "sarmfsw.h"
```

Include dependency graph for exceptions.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define exception_Handler(e)

    Exception handler asm caller.
- #define dump_stack()

    Dump stack asm caller.

Functions

- void stackDump (uint32_t stack[ ])

    prints contents of stack
- void HardFault_Handler_callback (uint32_t stack[ ])

    prints informations about current Hard Fault exception
- void Error_Handler_callback (uint32_t stack[ ])

    prints informations about current Hard Fault exception

5.2.1 Detailed Description

Debug tool and helpers declaration.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.2.2 Macro Definition Documentation

5.2.2.1 dump_stack

```
#define dump_stack( )
```

Value:

```
__asm(  "tst lr, #4 \r\n"       \
            "ite EQ \r\n"           \
            "mrseq r0, MSP \r\n"    \
            "mrsne r0, PSP \r\n"    \
            "b stackDump \r\n")
```

Dump stack asm caller.

5.2.2.2 exception_Handler

```
#define exception_Handler(
                e )
```

Value:

```
__asm(  "tst lr, #4 \r\n"                   \
            "ite EQ \r\n"                       \
            "mrseq r0, MSP \r\n"                \
            "mrsne r0, PSP \r\n"                \
            "b " #e "_Handler_callback \r\n")
```

Exception handler asm caller.

Note

The exception_Handler should be called with corresponding exception name **e** as parameter

5.2.3 Function Documentation

5.2.3.1 Error_Handler_callback()

```
void Error_Handler_callback (
            uint32_t stack[] )
```

prints informations about current Hard Fault exception

Parameters

in	*stack*	- pointer to stack address

Note

HardFault_Handler_callback should not be called directly use exception_Handler() which prepares pointer to current stack instead

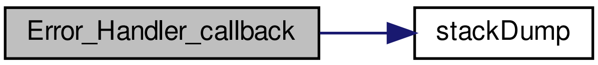
Warning

Depending how arm is fucked up, informations may not be printed, at least, you could inspect exception and stack through debug breakpoint

Returns

Never (anyways, arm fubared!)

Here is the call graph for this function:



5.2.3.2 HardFault_Handler_callback()

```
void HardFault_Handler_callback (
            uint32_t stack[ ] )
```

prints informations about current Hard Fault exception

Parameters

in	*stack*	- pointer to stack address

Note

HardFault_Handler_callback should not be called directly use exception_Handler() which prepares pointer to current stack instead

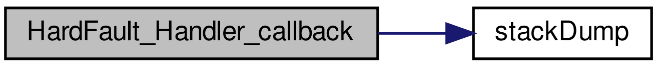
Warning

Depending how arm is fucked up, informations may not be printed, at least, you could inspect exception and stack through debug breakpoint

Returns

Never (anyways, arm fubared!)

Here is the call graph for this function:



5.2.3.3 stackDump()

```
void stackDump (
            uint32_t stack[] )
```

prints contents of stack

Parameters

in	*stack*	- pointer to stack address

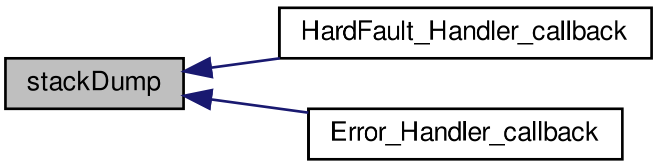
Note

stackDump should not be called directly, unless a particular stack is needed use dump_stack() which prepares pointer to current stack instead

Returns

Nothing

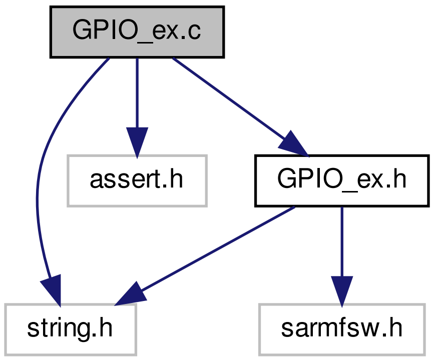
Here is the caller graph for this function:



5.3 GPIO_ex.c File Reference

Simple extension for GPIOs.

```
#include <string.h>
#include <assert.h>
#include "GPIO_ex.h"
```
Include dependency graph for GPIO_ex.c:



Macros

- #define MAX_PINS_PORT 16

Functions

- void GPIO_in_init (GPIO_in ∗in, GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin, uint16_t filter)

  Initialize GPIO_in instance.
- void GPIO_in_handler (GPIO_in ∗in)

  Handles GPIO_in read and treatment.
- FctERR str_GPIO_name (char ∗name, GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin)

  Get name from Port, Pin.

5.3.1 Detailed Description

Simple extension for GPIOs.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.3.2 Macro Definition Documentation

5.3.2.1 MAX_PINS_PORT

```
#define MAX_PINS_PORT 16
```

5.3.3 Function Documentation

5.3.3.1 GPIO_in_handler()

```
void GPIO_in_handler (
            GPIO_in * in )
```

Handles GPIO_in read and treatment.

Parameters

in,out	*in*	- input instance to handle

Returns

Nothing

5.3.3.2 GPIO_in_init()

```
void GPIO_in_init (
            GPIO_in * in,
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin,
            uint16_t filter )
```

Initialize GPIO_in instance.

Parameters

in,out	*in*	- input instance to initialize
in	*GPIOx*	- port to write to
in	*GPIO_Pin*	- pin to write to
in	*filter*	- input filtering time

Returns

Nothing

5.3.3.3 str_GPIO_name()

```
FctERR str_GPIO_name (
            char * name,
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin )
```

Get name from Port, Pin.

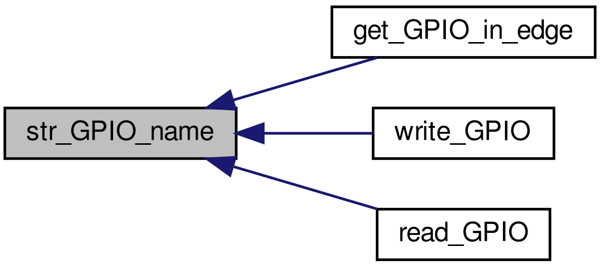
Parameters

in,out	*name*	- pointer to string for name
in	*GPIOx*	- port to write to
in	*GPIO_Pin*	- pin to write to

Returns

Error code

Here is the caller graph for this function:

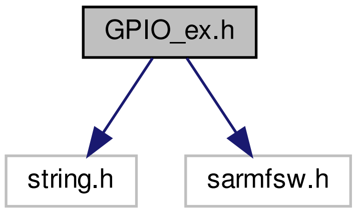


5.4 GPIO_ex.h File Reference

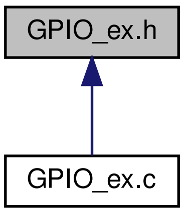
Simple extension for GPIOs.

```
#include <string.h>
#include "sarmfsw.h"
```

Include dependency graph for GPIO_ex.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct GPIO_in

  GPIO input structure.

Typedefs

- typedef enum ActOut eActOut
- typedef struct GPIO_in GPIO_in

Enumerations

- enum ActOut { Reset = 0, Set, Toggle }

  Logic output possible actions enumeration.

Functions

- void GPIO_in_init (GPIO_in ∗in, GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin, uint16_t filter)

  Initialize GPIO_in instance.
- void GPIO_in_handler (GPIO_in ∗in)

  Handles GPIO_in read and treatment.
- bool get_GPIO_in (GPIO_in ∗in)

  Get GPIO_in input value.
- bool get_GPIO_in_edge (GPIO_in ∗in)

  Get GPIO_in input edge.
- FctERR str_GPIO_name (char ∗name, GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin)

  Get name from Port, Pin.
- void write_GPIO (GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin, eActOut Act)

  Write GPIO.
- GPIO_PinState read_GPIO (GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin)

  Read GPIO.

5.4.1 Detailed Description

Simple extension for GPIOs.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.4.2 Typedef Documentation

5.4.2.1 eActOut

```
typedef enum ActOut eActOut
```

5.4.2.2 GPIO_in

```
typedef struct GPIO_in GPIO_in
```

5.4.3 Enumeration Type Documentation

5.4.3.1 ActOut

```
enum ActOut
```

Logic output possible actions enumeration.

Enumerator

Reset	Reset Output.
Set	Set Output.
Toggle	Toggle Output.

5.4.4 Function Documentation

5.4.4.1 get_GPIO_in()

```
bool get_GPIO_in (
            GPIO_in * in )  [inline]
```

Get GPIO_in input value.

Parameters

in	*in*	- input instance

Returns

Input value

5.4.4.2 get_GPIO_in_edge()

```
bool get_GPIO_in_edge (
            GPIO_in * in )  [inline]
```

Get GPIO_in input edge.

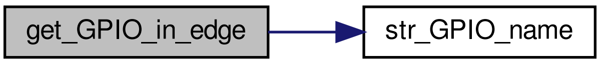
Parameters

in	*in*	- input instance

Returns

Input edge

Here is the call graph for this function:



5.4.4.3 GPIO_in_handler()

```
void GPIO_in_handler (
            GPIO_in * in )
```

Handles GPIO_in read and treatment.

Parameters

in,out	*in*	- input instance to handle

Returns

Nothing

5.4.4.4 GPIO_in_init()

```
void GPIO_in_init (
            GPIO_in * in,
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin,
            uint16_t filter )
```

Initialize GPIO_in instance.

Parameters

in,out	*in*	- input instance to initialize
in	*GPIOx*	- port to write to
in	*GPIO_Pin*	- pin to write to
in	*filter*	- input filtering time

Returns

Nothing

5.4.4.5 read_GPIO()

```
GPIO_PinState read_GPIO (
             GPIO_TypeDef * GPIOx,
             uint16_t GPIO_Pin )  [inline]
```

Read GPIO.

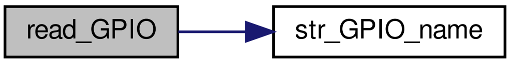
Parameters

in	*GPIOx*	- port to read from
in	*GPIO_Pin*	- pin to read from

Returns

Pin state

Here is the call graph for this function:



5.4.4.6 str_GPIO_name()

```
FctERR str_GPIO_name (
             char * name,
             GPIO_TypeDef * GPIOx,
             uint16_t GPIO_Pin )
```

Get name from Port, Pin.

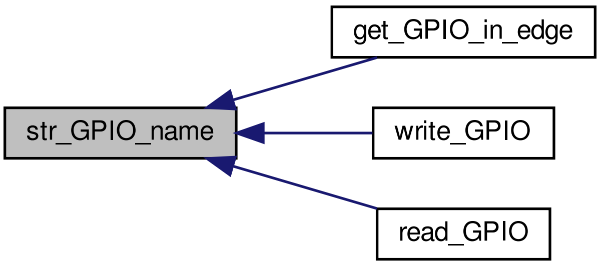
Parameters

in,out	*name*	- pointer to string for name
in	*GPIOx*	- port to write to
in	*GPIO_Pin*	- pin to write to

Returns

Error code

Here is the caller graph for this function:



5.4.4.7 write_GPIO()

```
void write_GPIO (
             GPIO_TypeDef * GPIOx,
             uint16_t GPIO_Pin,
             eActOut Act ) [inline]
```

Write GPIO.

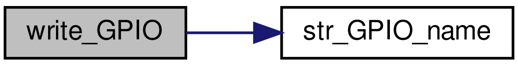
Parameters

in	*GPIOx*	- port to write to
in	*GPIO_Pin*	- pin to write to
in	*Act*	- type of write

Returns

Nothing

Here is the call graph for this function:

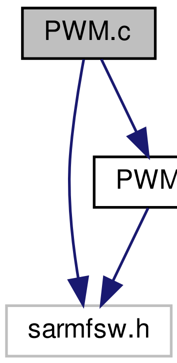


5.5 PWM.c File Reference

Straightforward PWM handling.

```
#include "sarmfsw.h"
#include "PWM.h"
```

Include dependency graph for PWM.c:



Functions

- HAL_StatusTypeDef init_TIM_Base (TIM_HandleTypeDef ∗pTim, uint32_t freq)

  Init TIM module and start interruptions.
- HAL_StatusTypeDef set_TIM_Freq (TIM_HandleTypeDef ∗pTim, uint32_t freq)

  Set TIM module frequency.
- HAL_StatusTypeDef init_PWM_Chan (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint16_t freq)

  Init TIM PWM module channel with frequency and starts the channel.
- HAL_StatusTypeDef set_PWM_Duty_Scaled (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint16_t duty, uint16_t scale)

  Set TIM module PWM duty cycle (scaled)
- FctERR logPWM_setPin (logicPWM ∗pPWM, GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin, bool polarity)

  Set channel pin & polarity for emulated PWM channel.
- FctERR logPWM_setFreq (logicPWM ∗pPWM, TIM_HandleTypeDef ∗pTim, uint16_t freq, uint16_t granularity)

  Set channel frequency for emulated PWM channel.
- FctERR logPWM_setDuty (logicPWM ∗pPWM, uint16_t val)

  Set new duty cycle for emulated PWM channel.
- FctERR logPWM_getFreq (uint16_t ∗freq, logicPWM ∗pPWM)

  Get channel frequency for emulated PWM channel.
- FctERR logPWM_getDutyCycle (float ∗duty, logicPWM ∗pPWM)

  Get channel Duty Cycle for emulated PWM channel.
- void logPWM_handler (logicPWM ∗pPWM)

  Handler for an emulated PWM channel.

5.5.1 Detailed Description

Straightforward PWM handling.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.5.2 Function Documentation

5.5.2.1 init_PWM_Chan()

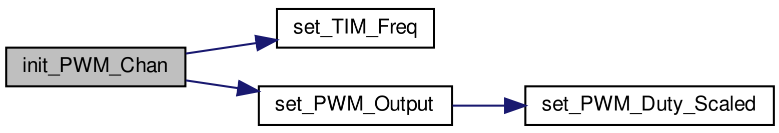
```
HAL_StatusTypeDef init_PWM_Chan (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t freq )
```

Init TIM PWM module channel with frequency and starts the channel.

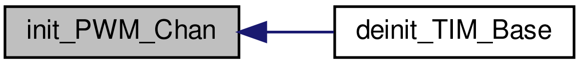
Parameters

in,out	*pTim*	- pointer to TIM instance for PWM generation
in	*chan*	- Channel to write
in	*freq*	- Desired PWM frequency

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.2 init_TIM_Base()

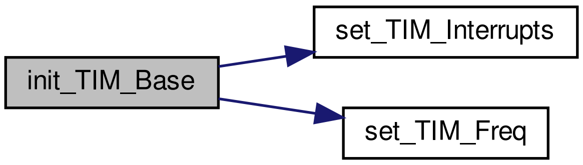
```
HAL_StatusTypeDef init_TIM_Base (
            TIM_HandleTypeDef * pTim,
            uint32_t freq )
```

Init TIM module and start interruptions.

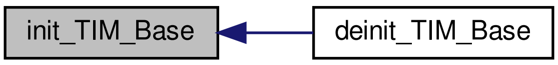
Parameters

in,out	*pTim*	- pointer to TIM instance
in	*freq*	- Desired TIM frequency

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.3 logPWM_getDutyCycle()

```
FctERR logPWM_getDutyCycle (
            float * duty,
            logicPWM * pPWM )
```

Get channel Duty Cycle for emulated PWM channel.

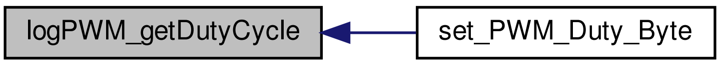
Parameters

in,out	*duty*	- pointer to duty cycle result
in,out	*pPWM*	- pointer to emulated PWM channel

Returns

Error code

Here is the caller graph for this function:



5.5.2.4 logPWM_getFreq()

```
FctERR logPWM_getFreq (
            uint16_t * freq,
            logicPWM * pPWM )
```

Get channel frequency for emulated PWM channel.

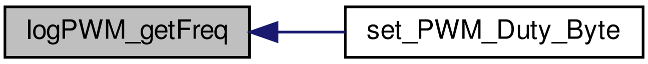
Parameters

in,out	*freq*	- pointer to frequency result
in,out	*pPWM*	- pointer to emulated PWM channel

Returns

Error code

Here is the caller graph for this function:



5.5.2.5 logPWM_handler()

```
void logPWM_handler (
            logicPWM * pPWM )
```

Handler for an emulated PWM channel.

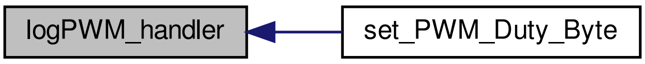
Warning

Shall be called directly from timer interrupt (HAL_TIM_PeriodElapsedCallback)

Parameters

in,out	*pPWM*	- pointer to emulated PWM channel

Here is the caller graph for this function:



5.5.2.6 logPWM_setDuty()

```
FctERR logPWM_setDuty (
            logicPWM * pPWM,
            uint16_t val )
```

Set new duty cycle for emulated PWM channel.

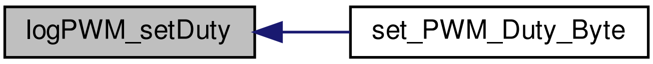
Parameters

`in,out`	*pPWM*	- pointer to emulated PWM channel
`in`	*val*	- Duty cycle to apply

Returns

Error code

Here is the caller graph for this function:



5.5.2.7 logPWM_setFreq()

```
FctERR logPWM_setFreq (
            logicPWM * pPWM,
            TIM_HandleTypeDef * pTim,
            uint16_t freq,
            uint16_t granularity )
```

Set channel frequency for emulated PWM channel.

Warning

For multiple PWMs on same timer with different frequencies, take care of init order (first configured channel will get TIM parameters precedence)

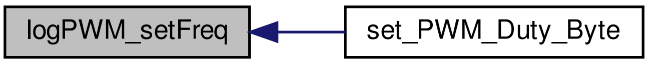
Parameters

`in,out`	*pPWM*	- pointer to emulated PWM channel
`in,out`	*pTim*	- pointer to TIM instance for Frequency computation
`in`	*freq*	- PWM frequency to apply
`in`	*granularity*	- PWM duty cycle granularity

Returns

Error code

Here is the caller graph for this function:



5.5.2.8 logPWM_setPin()

```
FctERR logPWM_setPin (
            logicPWM * pPWM,
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin,
            bool polarity )
```

Set channel pin & polarity for emulated PWM channel.

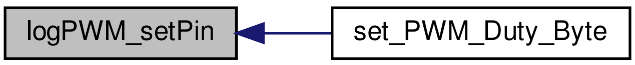
Parameters

`in`	*GPIOx*	- port for emulated PWM
`in`	*GPIO_Pin*	- pin for emulated PWM
`in,out`	*pPWM*	- pointer to emulated PWM channel
`in`	*polarity*	- 0: low polarity, 1: high polarity

Returns

Error code

Here is the caller graph for this function:



5.5.2.9 set_PWM_Duty_Scaled()

```
HAL_StatusTypeDef set_PWM_Duty_Scaled (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t duty,
            uint16_t scale )
```

Set TIM module PWM duty cycle (scaled)

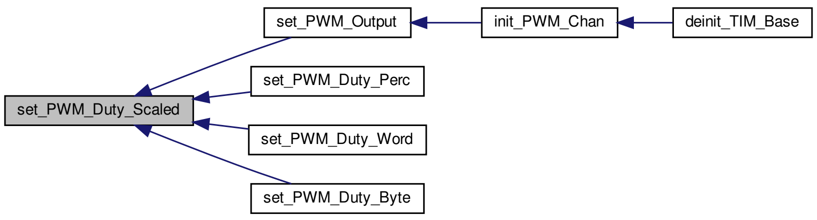
Parameters

`in,out`	*pTim*	- pointer to TIM instance for PWM generation
`in`	*chan*	- Channel to write
`in`	*duty*	- Scaled duty cycle value to write
`in`	*scale*	- Full scale value

Returns

HAL Status

Here is the caller graph for this function:



5.5.2.10 set_TIM_Freq()

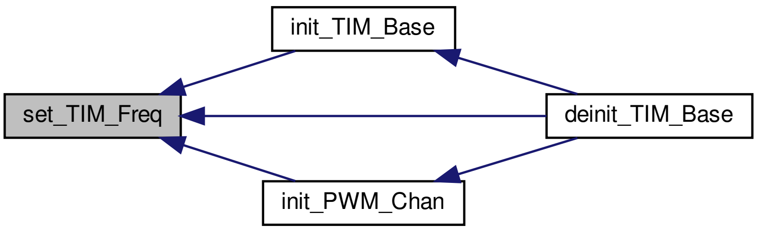
```
HAL_StatusTypeDef set_TIM_Freq (
            TIM_HandleTypeDef * pTim,
            uint32_t freq )
```

Set TIM module frequency.

Parameters

`in,out`	*pTim*	- pointer to TIM instance for Frequency computation
`in`	*freq*	- Desired TIM frequency

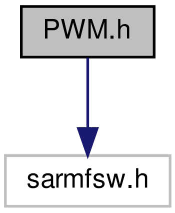
Here is the caller graph for this function:



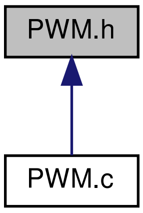
5.6 PWM.h File Reference

Straightforward PWM handling.

```
#include "sarmfsw.h"
```
Include dependency graph for PWM.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct logicPWM

  Software PWM on GPIO struct.

Typedefs

- typedef volatile struct logicPWM logicPWM

Functions

- HAL_StatusTypeDef set_TIM_Interrupts (TIM_HandleTypeDef ∗pTim, bool on)

  Start TIM module interrupts.
- HAL_StatusTypeDef deinit_TIM_Base (TIM_HandleTypeDef ∗pTim)

  De-Init TIM module and start interruptions.
- HAL_StatusTypeDef init_TIM_Base (TIM_HandleTypeDef ∗pTim, uint32_t freq)

  Init TIM module and start interruptions.
- HAL_StatusTypeDef set_TIM_Freq (TIM_HandleTypeDef ∗pTim, uint32_t freq)

  Set TIM module frequency.
- HAL_StatusTypeDef init_PWM_Chan (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint16_t freq)

  Init TIM PWM module channel with frequency and starts the channel.
- HAL_StatusTypeDef set_PWM_Output (TIM_HandleTypeDef ∗pTim, uint32_t chan, bool on)

  Set PWM channel output on/off.
- HAL_StatusTypeDef set_PWM_Duty_Scaled (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint16_t duty, uint16_t scale)

  Set TIM module PWM duty cycle (scaled)
- HAL_StatusTypeDef set_PWM_Duty_Perc (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint16_t duty)

  Set TIM module PWM duty cycle (percents)
- HAL_StatusTypeDef set_PWM_Duty_Word (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint16_t duty)

  Set TIM module PWM duty cycle (u16-bit value)
- HAL_StatusTypeDef set_PWM_Duty_Byte (TIM_HandleTypeDef ∗pTim, uint32_t chan, uint8_t duty)

  Set TIM module PWM duty cycle (u8-bit value)
- FctERR logPWM_setPin (logicPWM ∗pPWM, GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin, bool polarity)

  Set channel pin & polarity for emulated PWM channel.
- FctERR logPWM_setFreq (logicPWM ∗pPWM, TIM_HandleTypeDef ∗pTim, uint16_t freq, uint16_t granularity)

  Set channel frequency for emulated PWM channel.
- FctERR logPWM_setDuty (logicPWM ∗pPWM, uint16_t val)

  Set new duty cycle for emulated PWM channel.
- FctERR logPWM_getFreq (uint16_t ∗freq, logicPWM ∗pPWM)

  Get channel frequency for emulated PWM channel.
- FctERR logPWM_getDutyCycle (float ∗duty, logicPWM ∗pPWM)

  Get channel Duty Cycle for emulated PWM channel.
- void logPWM_handler (logicPWM ∗pPWM)

  Handler for an emulated PWM channel.

5.6.1 Detailed Description

Straightforward PWM handling.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

Warning

Shall work for all STM32 F families, L families not totally covered

5.6.2 Typedef Documentation

5.6.2.1 logicPWM

```
typedef volatile struct logicPWM logicPWM
```

5.6.3 Function Documentation

5.6.3.1 deinit_TIM_Base()

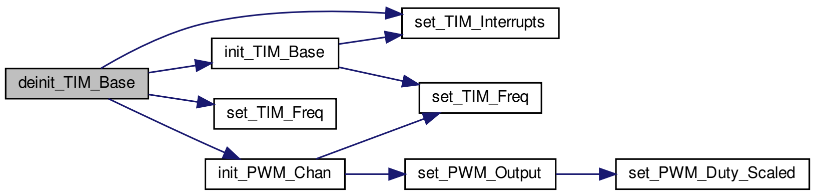
```
HAL_StatusTypeDef deinit_TIM_Base (
            TIM_HandleTypeDef * pTim )  [inline]
```

De-Init TIM module and start interruptions.

Parameters

`in,out`	*pTim*	- pointer to TIM instance

Here is the call graph for this function:



5.6.3.2 init_PWM_Chan()

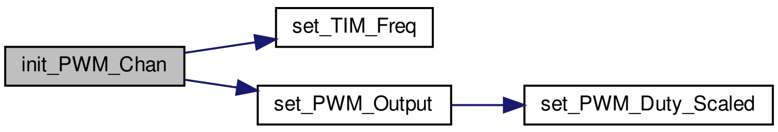
```
HAL_StatusTypeDef init_PWM_Chan (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t freq )
```

Init TIM PWM module channel with frequency and starts the channel.

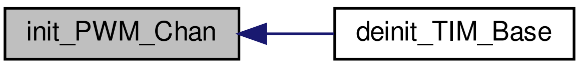
Parameters

`in,out`	*pTim*	- pointer to TIM instance for PWM generation
`in`	*chan*	- Channel to write
`in`	*freq*	- Desired PWM frequency

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.3 init_TIM_Base()

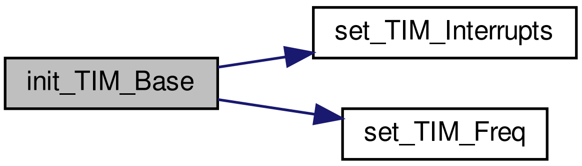
```
HAL_StatusTypeDef init_TIM_Base (
            TIM_HandleTypeDef * pTim,
            uint32_t freq )
```

Init TIM module and start interruptions.

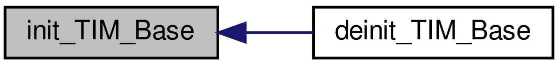
Parameters

`in,out`	*pTim*	- pointer to TIM instance
`in`	*freq*	- Desired TIM frequency

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.4 logPWM_getDutyCycle()

```
FctERR logPWM_getDutyCycle (
            float * duty,
            logicPWM * pPWM )
```

Get channel Duty Cycle for emulated PWM channel.

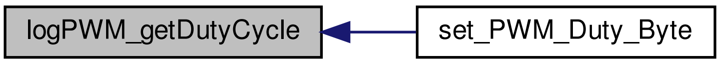
Parameters

`in,out`	*duty*	- pointer to duty cycle result
`in,out`	*pPWM*	- pointer to emulated PWM channel

Returns

Error code

Here is the caller graph for this function:



5.6.3.5 logPWM_getFreq()

```
FctERR logPWM_getFreq (
            uint16_t * freq,
            logicPWM * pPWM )
```

Get channel frequency for emulated PWM channel.

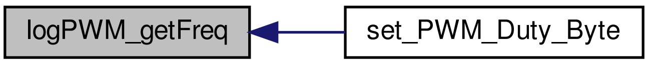
Parameters

`in,out`	*freq*	- pointer to frequency result
`in,out`	*pPWM*	- pointer to emulated PWM channel

Returns

Error code

Here is the caller graph for this function:



5.6.3.6 logPWM_handler()

```
void logPWM_handler (
             logicPWM * pPWM )
```

Handler for an emulated PWM channel.

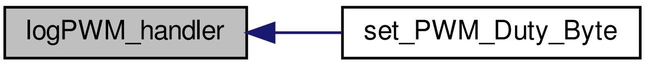
Warning

Shall be called directly from timer interrupt (HAL_TIM_PeriodElapsedCallback)

Parameters

in,out	*pPWM*	- pointer to emulated PWM channel

Here is the caller graph for this function:



5.6.3.7 logPWM_setDuty()

```
FctERR logPWM_setDuty (
             logicPWM * pPWM,
             uint16_t val )
```

Set new duty cycle for emulated PWM channel.

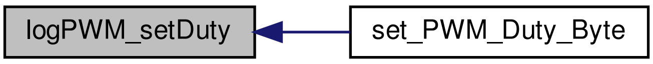
Parameters

in,out	*pPWM*	- pointer to emulated PWM channel
in	*val*	- Duty cycle to apply

Returns

Error code

Here is the caller graph for this function:



5.6.3.8 logPWM_setFreq()

```
FctERR logPWM_setFreq (
            logicPWM * pPWM,
            TIM_HandleTypeDef * pTim,
            uint16_t freq,
            uint16_t granularity )
```

Set channel frequency for emulated PWM channel.

Warning

For multiple PWMs on same timer with different frequencies, take care of init order (first configured channel will get TIM parameters precedence)

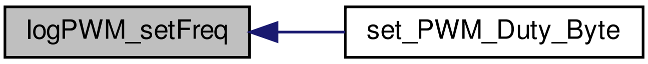
Parameters

in,out	*pPWM*	- pointer to emulated PWM channel
in,out	*pTim*	- pointer to TIM instance for Frequency computation
in	*freq*	- PWM frequency to apply
in	*granularity*	- PWM duty cycle granularity

Returns

Error code

Here is the caller graph for this function:



5.6.3.9 logPWM_setPin()

```
FctERR logPWM_setPin (
            logicPWM * pPWM,
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin,
            bool polarity )
```

Set channel pin & polarity for emulated PWM channel.

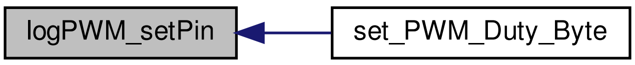
Parameters

`in`	*GPIOx*	- port for emulated PWM
`in`	*GPIO_Pin*	- pin for emulated PWM
`in,out`	*pPWM*	- pointer to emulated PWM channel
`in`	*polarity*	- 0: low polarity, 1: high polarity

Returns

Error code

Here is the caller graph for this function:



5.6.3.10 set_PWM_Duty_Byte()

```
HAL_StatusTypeDef set_PWM_Duty_Byte (
             TIM_HandleTypeDef * pTim,
             uint32_t chan,
             uint8_t duty )  [inline]
```

Set TIM module PWM duty cycle (u8-bit value)

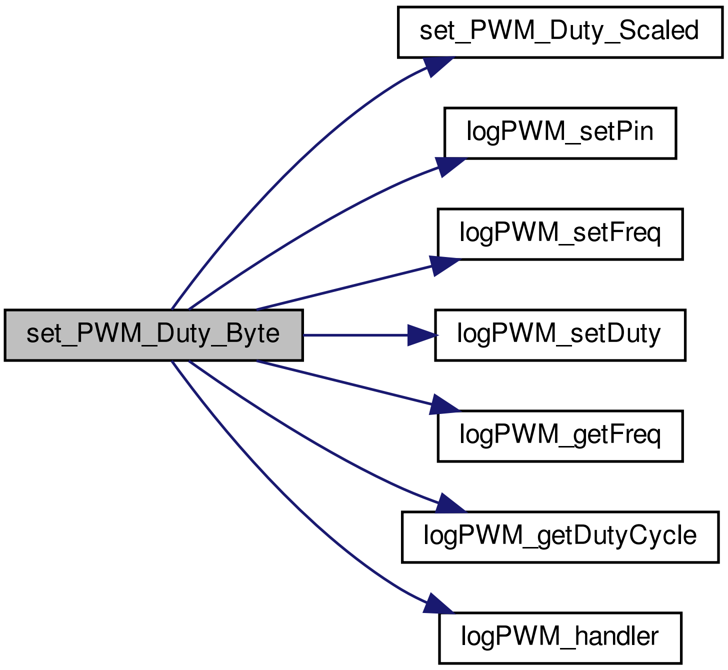
Parameters

`in,out`	*pTim*	- pointer to TIM instance for PWM generation
`in`	*chan*	- Channel to write
`in`	*duty*	- Scaled duty cycle value to write

Returns

HAL Status

Here is the call graph for this function:



5.6.3.11 set_PWM_Duty_Perc()

```
HAL_StatusTypeDef set_PWM_Duty_Perc (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t duty )  [inline]
```

Set TIM module PWM duty cycle (percents)

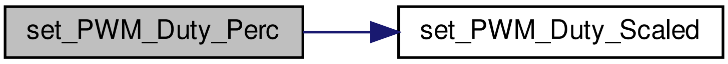
Parameters

`in,out`	*pTim*	- pointer to TIM instance for PWM generation
`in`	*chan*	- Channel to write
`in`	*duty*	- Scaled duty cycle value to write

Returns

HAL Status

Here is the call graph for this function:



5.6.3.12 set_PWM_Duty_Scaled()

```
HAL_StatusTypeDef set_PWM_Duty_Scaled (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t duty,
            uint16_t scale )
```

Set TIM module PWM duty cycle (scaled)

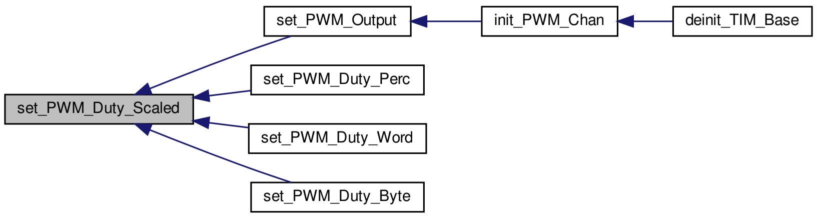
Parameters

`in,out`	*pTim*	- pointer to TIM instance for PWM generation
`in`	*chan*	- Channel to write
`in`	*duty*	- Scaled duty cycle value to write
`in`	*scale*	- Full scale value

Returns

HAL Status

Here is the caller graph for this function:



5.6.3.13 set_PWM_Duty_Word()

```
HAL_StatusTypeDef set_PWM_Duty_Word (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t duty )  [inline]
```

Set TIM module PWM duty cycle (u16-bit value)

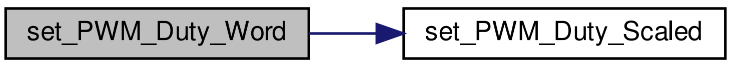
Parameters

in,out	*pTim*	- pointer to TIM instance for PWM generation
in	*chan*	- Channel to write
in	*duty*	- Scaled duty cycle value to write

Returns

HAL Status

Here is the call graph for this function:



5.6.3.14 set_PWM_Output()

```
HAL_StatusTypeDef set_PWM_Output (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            bool on )  [inline]
```

Set PWM channel output on/off.

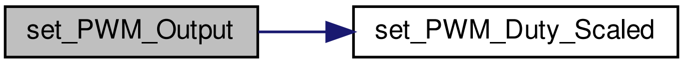
Parameters

`in,out`	*pTim*	- pointer to TIM instance for PWM generation
`in`	*chan*	- Channel to write
`in`	*on*	- Channel Output state 0: off, 1: on

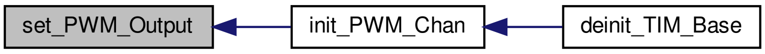
Returns

HAL Status

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.15 set_TIM_Freq()

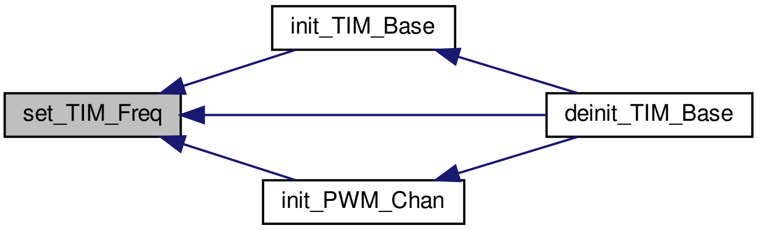
```
HAL_StatusTypeDef set_TIM_Freq (
            TIM_HandleTypeDef * pTim,
            uint32_t freq )
```

Set TIM module frequency.

Parameters

in,out	*pTim*	- pointer to TIM instance for Frequency computation
in	*freq*	- Desired TIM frequency

Here is the caller graph for this function:



5.6.3.16 set_TIM_Interrupts()

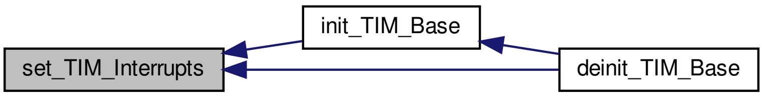
```
HAL_StatusTypeDef set_TIM_Interrupts (
            TIM_HandleTypeDef * pTim,
            bool on )  [inline]
```

Start TIM module interrupts.

Parameters

in,out	*pTim*	- pointer to TIM instance
in	*on*	- Time Interrupts 0: off, 1: on

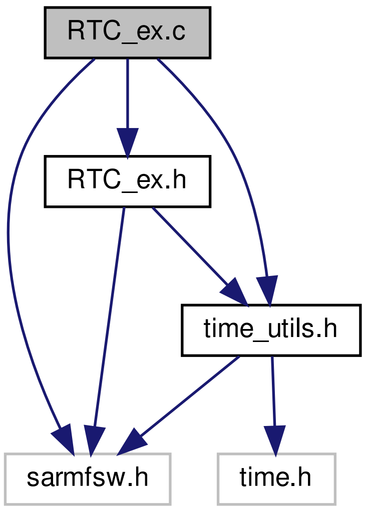
Here is the caller graph for this function:



5.7 RTC_ex.c File Reference

Basic RTC handling.

```
#include "sarmfsw.h"
#include "time_utils.h"
#include "RTC_ex.h"
```
Include dependency graph for RTC_ex.c:



Functions

- FctERR RTC_SetTime (DateTime ∗time_new)

  Sends new time to RTC peripheral.
- FctERR RTC_GetTime (DateTime ∗time_now)

  Get time from RTC peripheral.

5.7.1 Detailed Description

Basic RTC handling.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.7.2 Function Documentation

5.7.2.1 RTC_GetTime()

```
FctERR RTC_GetTime (
            DateTime * time_now )
```

Get time from RTC peripheral.

Parameters

in,out	*time_now*	- pointer to DateTime instance

Returns

FctERR - error code

5.7.2.2 RTC_SetTime()

```
FctERR RTC_SetTime (
            DateTime * time_new )
```

Sends new time to RTC peripheral.

Parameters

in	*time_new*	- pointer to DateTime instance

Returns

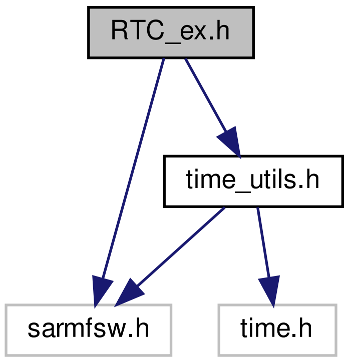
FctERR - error code

5.8 RTC_ex.h File Reference

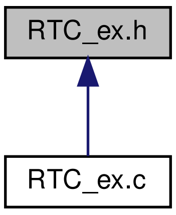
Basic RTC handling.

```
#include "sarmfsw.h"
#include "time_utils.h"
```

Include dependency graph for RTC_ex.h:



This graph shows which files directly or indirectly include this file:



Functions

- FctERR RTC_SetTime (DateTime ∗time_new)

  Sends new time to RTC peripheral.
- FctERR RTC_GetTime (DateTime ∗time_now)

  Get time from RTC peripheral.

5.8.1 Detailed Description

Basic RTC handling.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.8.2 Function Documentation

5.8.2.1 RTC_GetTime()

```
FctERR RTC_GetTime (
            DateTime * time_now )
```

Get time from RTC peripheral.

Parameters

in,out	*time_now*	- pointer to DateTime instance

Returns

FctERR - error code

5.8.2.2 RTC_SetTime()

```
FctERR RTC_SetTime (
            DateTime * time_new )
```

Sends new time to RTC peripheral.

Parameters

in	*time_new*	- pointer to DateTime instance

Returns

FctERR - error code

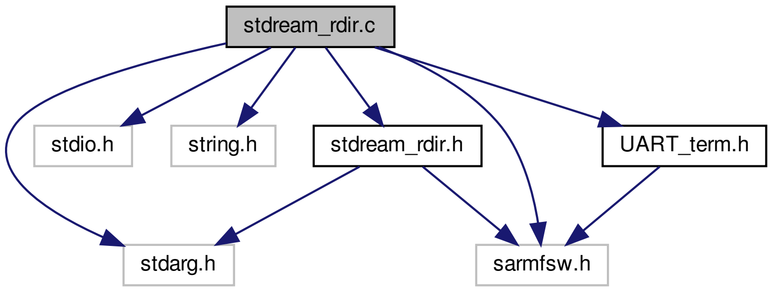
5.9 stdream_rdir.c File Reference

Stream redirection.

```
#include <stdarg.h>
#include <stdio.h>
```

```
#include <string.h>
#include "sarmfsw.h"
#include "UART_term.h"
#include "stdream_rdir.h"
```

Include dependency graph for stdream_rdir.c:



Functions

- void ITM_port_send (int port, const char ∗str, int len)

  Sends string to chosen ITM port.
- int printf_ITM (char ∗str,...)
- int vprintf_ITM (char ∗str, va_list args)
- int printf_rdir (char ∗str,...)
- int vprintf_rdir (char ∗str, va_list args)

Variables

- char dbg_msg_out [128] = ""

  stdream buffer for output
- char dbg_msg_in [32+1] = ""

  stdream buffer for input

5.9.1 Detailed Description

Stream redirection.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.9.2 Function Documentation

5.9.2.1 ITM_port_send()

```
void ITM_port_send (
             int port,
             const char * str,
             int len )
```

Sends string to chosen ITM port.

Get floating point number decimal part.

Parameters

in	*port*	- ITM port number
in	*str*	- pointer to string to send
in	*len*	- length of string

Returns

Nothing

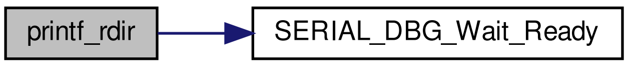
5.9.2.2 printf_ITM()

```
int printf_ITM (
             char * str,
              ... )
```

5.9.2.3 printf_rdir()

```
int printf_rdir (
             char * str,
              ... )
```

Here is the call graph for this function:



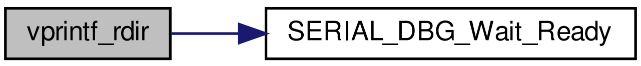
5.9.2.4 vprintf_ITM()

```
int vprintf_ITM (
             char * str,
             va_list args )
```

5.9.2.5 vprintf_rdir()

```
int vprintf_rdir (
             char * str,
             va_list args )
```

Here is the call graph for this function:



5.9.3 Variable Documentation

5.9.3.1 dbg_msg_in

```
char dbg_msg_in[32+1] = ""
```

stdream buffer for input

Warning

dbg_msg_in buffer for stdream is limited to **SZ_SERIAL_DBG_IN**

Note

dbg_msg_in is only related to UART_term

5.9.3.2 dbg_msg_out

```
char dbg_msg_out[128] = ""
```

stdream buffer for output

Warning

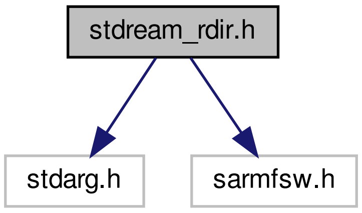
dbg_msg_out buffer for stdream is limited to **SZ_SERIAL_DBG_OUT** stdream buffer for output

5.10 stdream_rdir.h File Reference

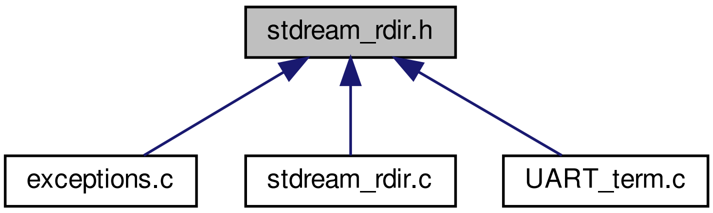
Stream redirection header.

```
#include <stdarg.h>
#include "sarmfsw.h"
```

Include dependency graph for stdream_rdir.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define printf printf_rdir

  Shadowing printf.
- #define vprintf vprintf_rdir

  Shadowing vprintf.
- #define SZ_SERIAL_DBG_OUT 128

  SERIAL DEBUG send buffer size.
- #define SZ_SERIAL_DBG_IN 32

  SERIAL DEBUG receive buffer size.

Functions

- void ITM_port_send (int port, const char ∗str, int len)

  Get floating point number decimal part.
- int printf_ITM (char ∗str,...)
- int vprintf_ITM (char ∗str, va_list args)
- int printf_rdir (char ∗str,...)
- int vprintf_rdir (char ∗str, va_list args)

Variables

- char dbg_msg_out [128]

  stdream buffer for output
- char dbg_msg_in [32+1]

  stdream buffer for input

5.10.1 Detailed Description

Stream redirection header.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

Note

define DBG_SERIAL in compiler defines with an UART instance to send printf likes strings to UART otherwise, define ITM_ENABLED in compiler defines for stings to be printed to ITM0 port

5.10.2 Macro Definition Documentation

5.10.2.1 printf

```
#define printf printf_rdir
```

Shadowing printf.

5.10.2.2 SZ_SERIAL_DBG_IN

```
#define SZ_SERIAL_DBG_IN 32
```

SERIAL DEBUG receive buffer size.

5.10.2.3 SZ_SERIAL_DBG_OUT

```
#define SZ_SERIAL_DBG_OUT 128
```

SERIAL DEBUG send buffer size.

5.10.2.4 vprintf

```
#define vprintf vprintf_rdir
```

Shadowing vprintf.

5.10.3 Function Documentation

5.10.3.1 ITM_port_send()

```
void ITM_port_send (
             int port,
             const char * str,
             int len )
```

Get floating point number decimal part.

Parameters

in	*port*	- ITM port number
in	*str*	- pointer to message to send
in	*len*	- length of message to send

Returns

Nothing

Get floating point number decimal part.

Parameters

in	*port*	- ITM port number
in	*str*	- pointer to string to send
in	*len*	- length of string

Returns

Nothing

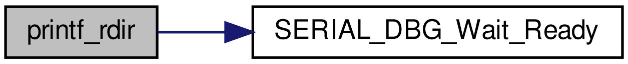
5.10.3.2 printf_ITM()

```
int printf_ITM (
             char * str,
              ...  )
```

5.10.3.3 printf_rdir()

```
int printf_rdir (
             char * str,
              ...  )
```

Here is the call graph for this function:



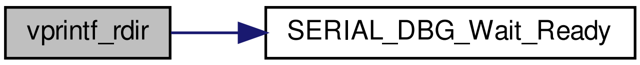
5.10.3.4 vprintf_ITM()

```
int vprintf_ITM (
             char * str,
             va_list args )
```

5.10.3.5 vprintf_rdir()

```
int vprintf_rdir (
             char * str,
             va_list args )
```

Here is the call graph for this function:



5.10.4 Variable Documentation

5.10.4.1 dbg_msg_in

```
char dbg_msg_in[32+1]
```

stdream buffer for input

Warning

dbg_msg_in buffer for stdream is limited to **SZ_SERIAL_DBG_IN**

Note

dbg_msg_in is only related to UART_term

5.10.4.2 dbg_msg_out

```
char dbg_msg_out[128]
```

stdream buffer for output

Warning

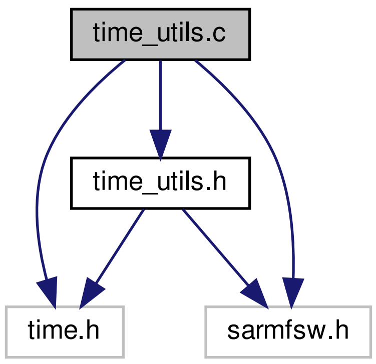
dbg_msg_out buffer for stdream is limited to **SZ_SERIAL_DBG_OUT** stdream buffer for output

5.11 time_utils.c File Reference

Time related utilities.

```
#include <time.h>
#include "sarmfsw.h"
#include "time_utils.h"
```

Include dependency graph for time_utils.c:



Functions

- DateTime time_t2DateTime (time_t time)

  Convert time_t to DateTime.
- time_t DateTime2time_t (DateTime ∗time)

  Convert DateTime to time_t.
- DateTime diffDateTime (DateTime ∗time2, DateTime ∗time1)

  Calculate DateTime difference.

5.11.1 Detailed Description

Time related utilities.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.11.2 Function Documentation

5.11.2.1 DateTime2time_t()

```
time_t DateTime2time_t (
            DateTime * time )
```

Convert DateTime to time_t.

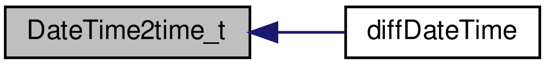
Parameters

in	*time*	- DateTime representation (broken down time)

Returns

time_t representation

Here is the caller graph for this function:



5.11.2.2 diffDateTime()

```
DateTime diffDateTime (
            DateTime * time2,
            DateTime * time1 )
```

Calculate DateTime difference.

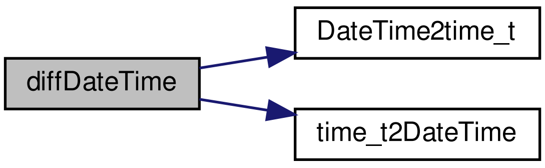
Parameters

in	*time2*	- pointer to closest DateTime representation (broken down time)
in	*time1*	- pointer to oldest DateTime representation (broken down time)

Returns

DateTime difference

Here is the call graph for this function:



5.11.2.3 time_t2DateTime()

```
DateTime time_t2DateTime (
            time_t time )
```

Convert time_t to DateTime.

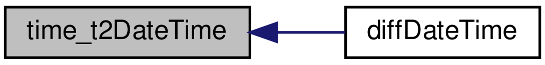
Parameters

in	*time*	- time_t representation

Returns

Broken down time representation (DateTime)

Here is the caller graph for this function:

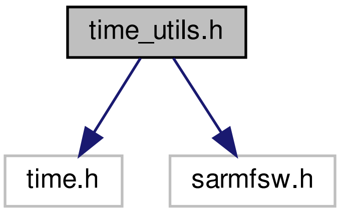


5.12 time_utils.h File Reference

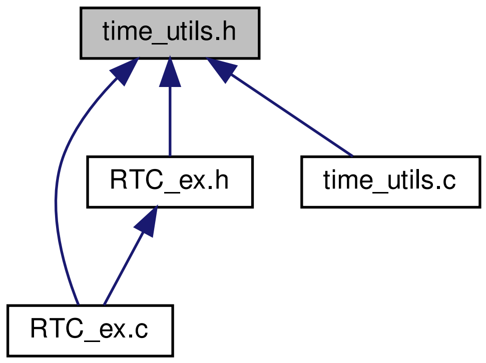
Time related utilities.

```
#include <time.h>
#include "sarmfsw.h"
```

Include dependency graph for time_utils.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct DateTime

  Basic Date & Time struct.

Typedefs

- typedef struct DateTime DateTime

Functions

- DateTime time_t2DateTime (time_t time)

  Convert time_t to DateTime.
- time_t DateTime2time_t (DateTime ∗time)

  Convert DateTime to time_t.
- DateTime diffDateTime (DateTime ∗time2, DateTime ∗time1)

  Calculate DateTime difference.

5.12.1 Detailed Description

Time related utilities.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.12.2 Typedef Documentation

5.12.2.1 DateTime

```
typedef struct DateTime DateTime
```

5.12.3 Function Documentation

5.12.3.1 DateTime2time_t()

```
time_t DateTime2time_t (
            DateTime * time )
```

Convert DateTime to time_t.

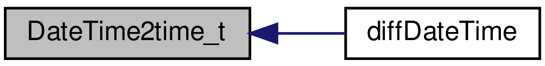
Parameters

in	*time*	- DateTime representation (broken down time)

Returns

time_t representation

Here is the caller graph for this function:



5.12.3.2 diffDateTime()

```
DateTime diffDateTime (
            DateTime * time2,
            DateTime * time1 )
```

Calculate DateTime difference.

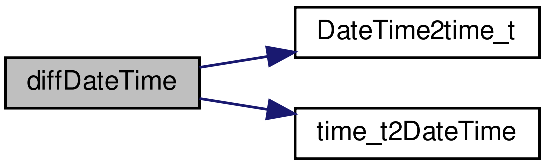
Parameters

in	*time2*	- pointer to closest DateTime representation (broken down time)
in	*time1*	- pointer to oldest DateTime representation (broken down time)

Returns

DateTime difference

Here is the call graph for this function:



5.12.3.3 time_t2DateTime()

```
DateTime time_t2DateTime (
            time_t time )
```

Convert time_t to DateTime.

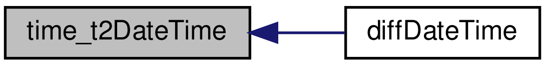
Parameters

in	*time*	- time_t representation

Returns

Broken down time representation (DateTime)

Here is the caller graph for this function:

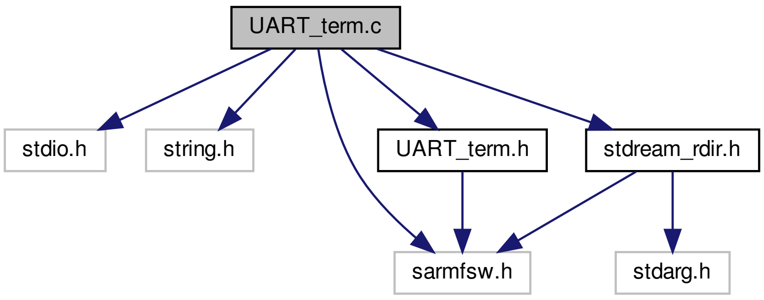


5.13 UART_term.c File Reference

UART terminal.

```
#include <stdio.h>
#include <string.h>
#include "sarmfsw.h"
#include "stdream_rdir.h"
#include "UART_term.h"
```

Include dependency graph for UART_term.c:



Functions

- FctERR SERIAL_DBG_Launch_It_Rx (UART_HandleTypeDef ∗huart)

  Start UART SERIAL DEBUG Rx interruptions.
- FctERR SERIAL_DBG_Flush_RxBuf (UART_HandleTypeDef ∗huart)

  Clear buffer in used for SERIAL DEBUG.
- FctERR SERIAL_DBG_Message_Handler (const char ∗msg, uint8_t len)

  Treat fully received message.
- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)

  Rx Transfer completed callback.
- void HAL_UART_TxCpltCallback (UART_HandleTypeDef ∗huart)

  Tx Transfer completed callback (clear uart_out buffer)

Variables

- char breakout_char = '!'

  breakout char (message complete)
- UART_HandleTypeDef ∗ dbg_uart = 1

  Instance of UART debug terminal.

5.13.1 Detailed Description

UART terminal.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

5.13.2 Function Documentation

5.13.2.1 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
             UART_HandleTypeDef * huart )
```

Rx Transfer completed callback.

Parameters

huart	UART handle.

Return values

None	

5.13.2.2 HAL_UART_TxCpltCallback()

```
void HAL_UART_TxCpltCallback (
             UART_HandleTypeDef * huart )
```

Tx Transfer completed callback (clear uart_out buffer)

Parameters

huart	- UART handle

Return values

None	

5.13.2.3 SERIAL_DBG_Flush_RxBuf()

```
FctERR SERIAL_DBG_Flush_RxBuf (
            UART_HandleTypeDef * huart )
```

Clear buffer in used for SERIAL DEBUG.

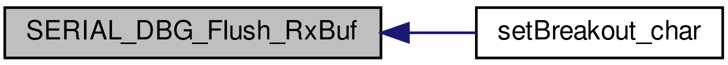
Parameters

in	*huart*	- UART handle (reserved for future use if needed)

Returns

Error code

Here is the caller graph for this function:



5.13.2.4 SERIAL_DBG_Launch_It_Rx()

```
FctERR SERIAL_DBG_Launch_It_Rx (
            UART_HandleTypeDef * huart )
```

Start UART SERIAL DEBUG Rx interruptions.

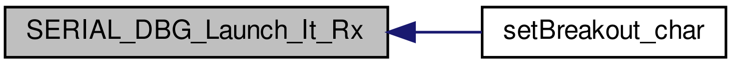
Parameters

in	*huart*	- UART handle

Returns

Error code

Here is the caller graph for this function:



5.13.2.5 SERIAL_DBG_Message_Handler()

```
FctERR SERIAL_DBG_Message_Handler (
             const char * msg,
             uint8_t len )
```

Treat fully received message.

Weak Functions This function is implemented as weak to be implemented in projects (weak one only prints & flushes the buffer)

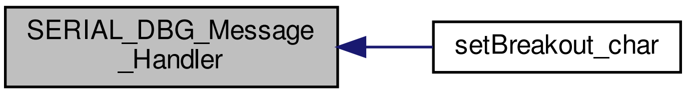
Parameters

in	*msg*	- pointer to received message
in	*len*	- received message length

Returns

Error code

Here is the caller graph for this function:



5.13.3 Variable Documentation

5.13.3.1 breakout_char

```
char breakout_char = '!'
```

breakout char (message complete)

Note

Default user breakout char set to '!' and '\r' is built-in default breakout char

5.13.3.2 dbg_uart

```
UART_HandleTypeDef* dbg_uart = 1
```

Instance of UART debug terminal.

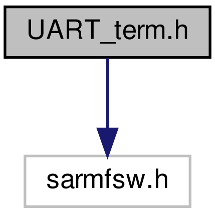
UART debug terminal instance.

5.14 UART_term.h File Reference

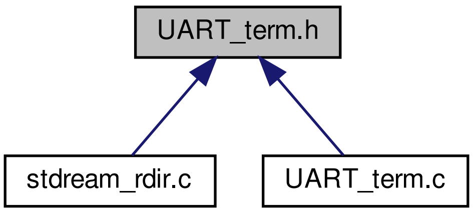
UART terminal header.

```
#include "sarmfsw.h"
```

Include dependency graph for UART_term.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define STDREAM__UART_TX_IT

    To be defined to send to uart using interrupts.

Functions

- char getBreakout_char (void)

    Get UART Rx breakout character.
- void setBreakout_char (char breakout)

    Set a new breakout character.
- FctERR SERIAL_DBG_Launch_It_Rx (UART_HandleTypeDef ∗huart)

    Start UART SERIAL DEBUG Rx interruptions.
- FctERR SERIAL_DBG_Flush_RxBuf (UART_HandleTypeDef ∗huart)

    Clear buffer in used for SERIAL DEBUG.
- FctERR SERIAL_DBG_Message_Handler (const char ∗msg, uint8_t len)

    Treat fully received message.
- void SERIAL_DBG_Wait_Ready (UART_HandleTypeDef ∗huart)

    Waiting for UART global state to be ready for next transmission.
- HAL_StatusTypeDef SERIAL_DBG_Send (UART_HandleTypeDef ∗huart, char ∗str, int len)

    Sends string to UART.

Variables

- char breakout_char

    breakout char (message complete)
- UART_HandleTypeDef ∗ dbg_uart

    UART debug terminal instance.

5.14.1 Detailed Description

UART terminal header.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

Note

define DBG_SERIAL in compiler defines with an UART instance to send printf likes strings to UART

5.14.2 Macro Definition Documentation

5.14.2.1 STDREAM__UART_TX_IT

```
#define STDREAM__UART_TX_IT
```

To be defined to send to uart using interrupts.

5.14.3 Function Documentation

5.14.3.1 getBreakout_char()

```
char getBreakout_char (
             void  )  [inline]
```

Get UART Rx breakout character.

Returns

Breakout character

5.14.3.2 SERIAL_DBG_Flush_RxBuf()

```
FctERR SERIAL_DBG_Flush_RxBuf (
             UART_HandleTypeDef * huart )
```

Clear buffer in used for SERIAL DEBUG.

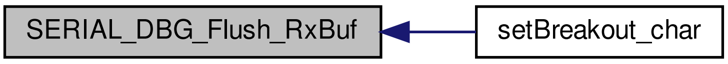
Parameters

in	*huart*	- UART handle (reserved for future use if needed)

Returns

Error code

Here is the caller graph for this function:



5.14.3.3 SERIAL_DBG_Launch_It_Rx()

```
FctERR SERIAL_DBG_Launch_It_Rx (
            UART_HandleTypeDef * huart )
```

Start UART SERIAL DEBUG Rx interruptions.

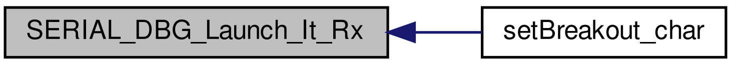
Parameters

in	*huart*	- UART handle

Returns

Error code

Here is the caller graph for this function:



5.14.3.4 SERIAL_DBG_Message_Handler()

```
FctERR SERIAL_DBG_Message_Handler (
            const char * msg,
            uint8_t len )
```

Treat fully received message.

Weak Functions This function is implemented as weak to be implemented in projects (weak one only prints & flushes the buffer)

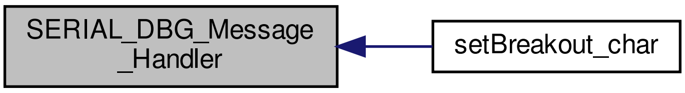
Parameters

in	*msg*	- pointer to received message
in	*len*	- received message length

Returns

Error code

Here is the caller graph for this function:



5.14.3.5 SERIAL_DBG_Send()

```
HAL_StatusTypeDef SERIAL_DBG_Send (
             UART_HandleTypeDef * huart,
             char * str,
             int len ) [inline]
```

Sends string to UART.

Parameters

in	*huart*	- UART handle
in	*str*	- pointer to string to send
in	*len*	- length of string

Returns

HAL Status

5.14.3.6 SERIAL_DBG_Wait_Ready()

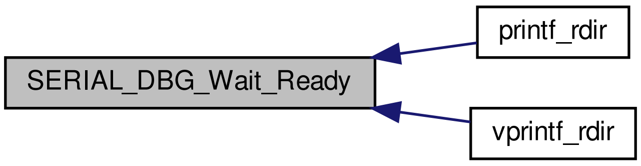
```
void SERIAL_DBG_Wait_Ready (
             UART_HandleTypeDef * huart ) [inline]
```

Waiting for UART global state to be ready for next transmission.

Parameters

in	*huart*	- UART handle

Here is the caller graph for this function:



5.14.3.7 setBreakout_char()

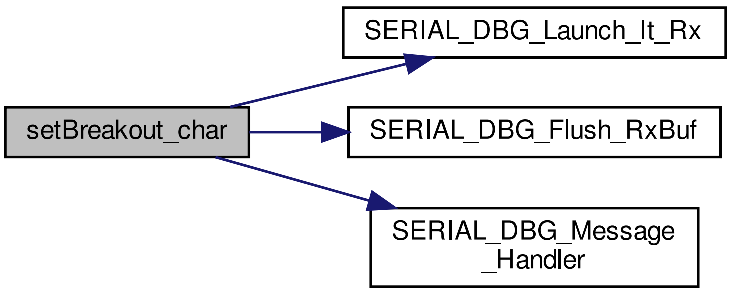
```
void setBreakout_char (
             char breakout )  [inline]
```

Set a new breakout character.

Parameters

in	*breakout*	- new breakout character

Here is the call graph for this function:



5.14.4 Variable Documentation

5.14.4.1 breakout_char

```
char breakout_char
```

breakout char (message complete)

Note

Default user breakout char set to '!' and '\r' is built-in default breakout char

5.14.4.2 dbg_uart

```
UART_HandleTypeDef* dbg_uart
```

UART debug terminal instance.

UART debug terminal instance.

Index

ActOut
  GPIO_ex.h, 21

breakout_char
  UART_term.c, 69
  UART_term.h, 77

cfg
  GPIO_in, 5
  logicPWM, 8
cntr
  logicPWM, 8

DateTime, 3
  Day, 4
  Hours, 4
  Minutes, 4
  Month, 4
  Seconds, 4
  time_utils.h, 64
  Weekday, 4
  Year, 4
DateTime2time_t
  time_utils.c, 60
  time_utils.h, 64
Day
  DateTime, 4
dbg_msg_in
  stdream_rdir.c, 54
  stdream_rdir.h, 59
dbg_msg_out
  stdream_rdir.c, 54
  stdream_rdir.h, 59
dbg_uart
  UART_term.c, 70
  UART_term.h, 77
deinit_TIM_Base
  PWM.h, 36
diffDateTime
  time_utils.c, 61
  time_utils.h, 64
done
  GPIO_in, 6
dump_stack
  exceptions.h, 14
duty
  logicPWM, 8

eActOut
  GPIO_ex.h, 21
edge
  GPIO_in, 6
Error_Handler_callback
  exceptions.c, 10
  exceptions.h, 14
exception_Handler
  exceptions.h, 14
exceptions.c, 9
  Error_Handler_callback, 10
  HardFault_Handler_callback, 11
  stackDump, 11
exceptions.h, 12
  dump_stack, 14
  Error_Handler_callback, 14
  exception_Handler, 14
  HardFault_Handler_callback, 15
  stackDump, 16

filt
  GPIO_in, 6

GPIO_Pin
  GPIO_in, 6
  logicPWM, 8
GPIO_ex.c, 17
  GPIO_in_handler, 18
  GPIO_in_init, 18
  MAX_PINS_PORT, 18
  str_GPIO_name, 19
GPIO_ex.h, 19
  ActOut, 21
  eActOut, 21
  GPIO_in, 21
  GPIO_in_handler, 23
  GPIO_in_init, 23
  get_GPIO_in, 22
  get_GPIO_in_edge, 22
  read_GPIO, 24
  str_GPIO_name, 24
  write_GPIO, 25
GPIO_in, 5
  cfg, 5
  done, 6
  edge, 6
  filt, 6
  GPIO_Pin, 6
  GPIO_ex.h, 21
  GPIOx, 6
  hIn, 6
  in, 6
  mem, 7
GPIO_in_handler
  GPIO_ex.c, 18
  GPIO_ex.h, 23
GPIO_in_init
  GPIO_ex.c, 18
  GPIO_ex.h, 23
GPIOx
  GPIO_in, 6
  logicPWM, 8
get_GPIO_in
  GPIO_ex.h, 22

get_GPIO_in_edge
    GPIO_ex.h, 22
getBreakout_char
    UART_term.h, 72

HAL_UART_RxCpltCallback
    UART_term.c, 67
HAL_UART_TxCpltCallback
    UART_term.c, 67
hIn
    GPIO_in, 6
HardFault_Handler_callback
    exceptions.c, 11
    exceptions.h, 15
Hours
    DateTime, 4

ITM_port_send
    stdream_rdir.c, 53
    stdream_rdir.h, 57
in
    GPIO_in, 6
init_PWM_Chan
    PWM.c, 27
    PWM.h, 37
init_TIM_Base
    PWM.c, 28
    PWM.h, 38

logPWM_getDutyCycle
    PWM.c, 29
    PWM.h, 38
logPWM_getFreq
    PWM.c, 29
    PWM.h, 39
logPWM_handler
    PWM.c, 30
    PWM.h, 39
logPWM_setDuty
    PWM.c, 30
    PWM.h, 40
logPWM_setFreq
    PWM.c, 31
    PWM.h, 41
logPWM_setPin
    PWM.c, 32
    PWM.h, 42
logicPWM, 7
    cfg, 8
    cntr, 8
    duty, 8
    GPIO_Pin, 8
    GPIOx, 8
    pTim, 9
    PWM.h, 36
    per, 8
    polarity, 8
    tim_freq, 9

MAX_PINS_PORT
    GPIO_ex.c, 18
mem
    GPIO_in, 7
Minutes
    DateTime, 4
Month
    DateTime, 4

pTim
    logicPWM, 9
PWM.c, 26
    init_PWM_Chan, 27
    init_TIM_Base, 28
    logPWM_getDutyCycle, 29
    logPWM_getFreq, 29
    logPWM_handler, 30
    logPWM_setDuty, 30
    logPWM_setFreq, 31
    logPWM_setPin, 32
    set_PWM_Duty_Scaled, 32
    set_TIM_Freq, 33
PWM.h, 34
    deinit_TIM_Base, 36
    init_PWM_Chan, 37
    init_TIM_Base, 38
    logPWM_getDutyCycle, 38
    logPWM_getFreq, 39
    logPWM_handler, 39
    logPWM_setDuty, 40
    logPWM_setFreq, 41
    logPWM_setPin, 42
    logicPWM, 36
    set_PWM_Duty_Byte, 42
    set_PWM_Duty_Perc, 43
    set_PWM_Duty_Scaled, 44
    set_PWM_Duty_Word, 45
    set_PWM_Output, 45
    set_TIM_Freq, 46
    set_TIM_Interrupts, 47
per
    logicPWM, 8
polarity
    logicPWM, 8
printf
    stdream_rdir.h, 56
printf_ITM
    stdream_rdir.c, 53
    stdream_rdir.h, 58
printf_rdir
    stdream_rdir.c, 53
    stdream_rdir.h, 58

RTC_GetTime
    RTC_ex.c, 49
    RTC_ex.h, 51
RTC_SetTime
    RTC_ex.c, 49
    RTC_ex.h, 51

RTC_ex.c, 47
 RTC_GetTime, 49
 RTC_SetTime, 49
RTC_ex.h, 49
 RTC_GetTime, 51
 RTC_SetTime, 51
read_GPIO
 GPIO_ex.h, 24

SERIAL_DBG_Flush_RxBuf
 UART_term.c, 68
 UART_term.h, 72
SERIAL_DBG_Launch_It_Rx
 UART_term.c, 68
 UART_term.h, 73
SERIAL_DBG_Message_Handler
 UART_term.c, 69
 UART_term.h, 73
SERIAL_DBG_Send
 UART_term.h, 75
SERIAL_DBG_Wait_Ready
 UART_term.h, 75
STDREAM__UART_TX_IT
 UART_term.h, 72
SZ_SERIAL_DBG_IN
 stdream_rdir.h, 56
SZ_SERIAL_DBG_OUT
 stdream_rdir.h, 57
Seconds
 DateTime, 4
set_PWM_Duty_Byte
 PWM.h, 42
set_PWM_Duty_Perc
 PWM.h, 43
set_PWM_Duty_Scaled
 PWM.c, 32
 PWM.h, 44
set_PWM_Duty_Word
 PWM.h, 45
set_PWM_Output
 PWM.h, 45
set_TIM_Freq
 PWM.c, 33
 PWM.h, 46
set_TIM_Interrupts
 PWM.h, 47
setBreakout_char
 UART_term.h, 76
stackDump
 exceptions.c, 11
 exceptions.h, 16
stdream_rdir.c, 51
 dbg_msg_in, 54
 dbg_msg_out, 54
 ITM_port_send, 53
 printf_ITM, 53
 printf_rdir, 53
 vprintf_ITM, 53
 vprintf_rdir, 54
stdream_rdir.h, 55
 dbg_msg_in, 59
 dbg_msg_out, 59
 ITM_port_send, 57
 printf, 56
 printf_ITM, 58
 printf_rdir, 58
 SZ_SERIAL_DBG_IN, 56
 SZ_SERIAL_DBG_OUT, 57
 vprintf, 57
 vprintf_ITM, 58
 vprintf_rdir, 58
str_GPIO_name
 GPIO_ex.c, 19
 GPIO_ex.h, 24

tim_freq
 logicPWM, 9
time_t2DateTime
 time_utils.c, 61
 time_utils.h, 65
time_utils.c, 59
 DateTime2time_t, 60
 diffDateTime, 61
 time_t2DateTime, 61
time_utils.h, 62
 DateTime, 64
 DateTime2time_t, 64
 diffDateTime, 64
 time_t2DateTime, 65

UART_term.c, 66
 breakout_char, 69
 dbg_uart, 70
 HAL_UART_RxCpltCallback, 67
 HAL_UART_TxCpltCallback, 67
 SERIAL_DBG_Flush_RxBuf, 68
 SERIAL_DBG_Launch_It_Rx, 68
 SERIAL_DBG_Message_Handler, 69
UART_term.h, 70
 breakout_char, 77
 dbg_uart, 77
 getBreakout_char, 72
 SERIAL_DBG_Flush_RxBuf, 72
 SERIAL_DBG_Launch_It_Rx, 73
 SERIAL_DBG_Message_Handler, 73
 SERIAL_DBG_Send, 75
 SERIAL_DBG_Wait_Ready, 75
 STDREAM__UART_TX_IT, 72
 setBreakout_char, 76

vprintf
 stdream_rdir.h, 57
vprintf_ITM
 stdream_rdir.c, 53
 stdream_rdir.h, 58
vprintf_rdir
 stdream_rdir.c, 54
 stdream_rdir.h, 58

Weekday
    DateTime, 4
write_GPIO
    GPIO_ex.h, 25

Year
    DateTime, 4