HARMcksL: ARM HAL toolbox (yet STM32 oriented)

1.0

Generated by Doxygen 1.8.13

Contents

1 Class Index

1.1 Class List

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

GPIO_in
GPIO input structure **2**

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

3 Class Documentation

3.1 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

3.1.1 Detailed Description

GPIO input structure.

3.1.2 Member Data Documentation

3.1.2.1 cfg

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

3.1.2.2 done

```
bool GPIO_in::done
```

State change done.

3.1.2.3 edge

```
eEdge GPIO_in::edge
```

Input edge.

3.1.2.4 filt

```
uint16_t GPIO_in::filt
```

Filter time (ms)

3.1.2.5 GPIO_Pin

```
uint16_t GPIO_in::GPIO_Pin
```

HAL GPIO pin.

3.1.2.6 GPIOx

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

HAL GPIO instance.

3.1.2.7 hIn

```
uint32_t GPIO_in::hIn
```

Filter time.

3.1.2.8 in

```
bool GPIO_in::in
```

Input value.

3.1.2.9 mem

```
bool GPIO_in::mem
```

Memo value.

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

- GPIO_ex.h

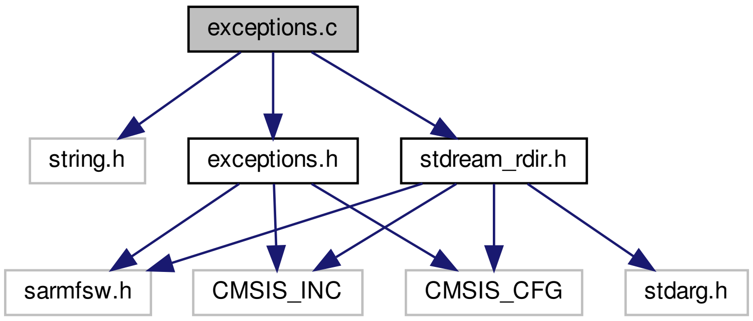
4 File Documentation

4.1 exceptions.c File Reference

Debug tool helpers functions.

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

4.1.1 Detailed Description

Debug tool helpers functions.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.1.2 Function Documentation

4.1.2.1 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

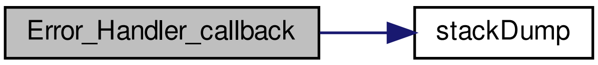
Warning

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:



4.1.2.2 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

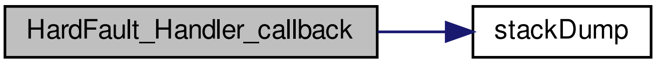
Warning

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:



4.1.2.3 stackDump()

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

prints contents of stack

Parameters

in	*stack*	- pointer to stack address

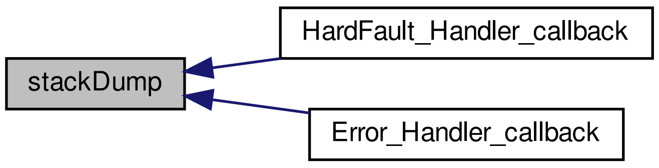
Note

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:

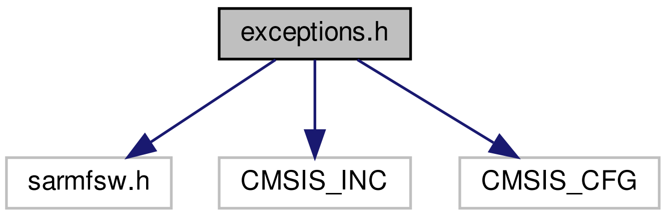


4.2 exceptions.h File Reference

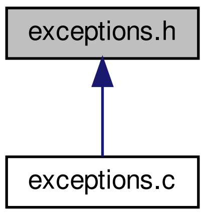
Debug tool and helpers declaration.

```
#include "sarmfsw.h"
#include <CMSIS_INC>
#include <CMSIS_CFG>
```

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

4.2.1 Detailed Description

Debug tool and helpers declaration.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.2.2 Macro Definition Documentation

4.2.2.1 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.

4.2.2.2 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

4.2.3 Function Documentation

4.2.3.1 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

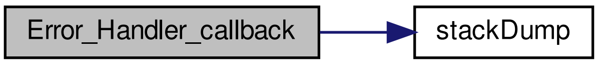
Warning

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:



4.2.3.2 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

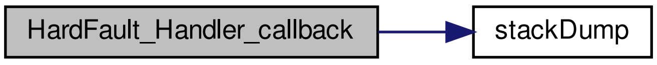
Warning

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:



4.2.3.3 stackDump()

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

prints contents of stack

Parameters

in	*stack*	- pointer to stack address

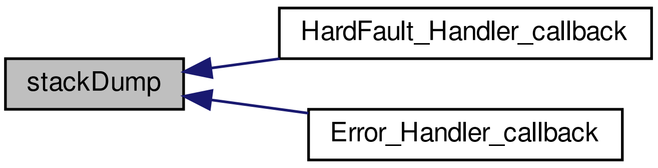
Note

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:

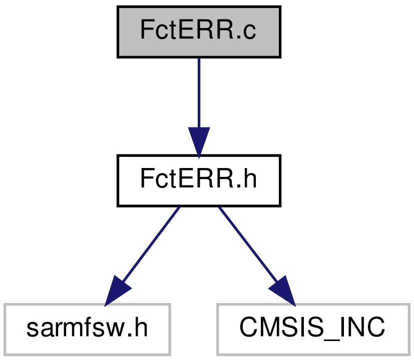


4.3 FctERR.c File Reference

errors to SMFSW FctERR code

```
#include "FctERR.h"
```

Include dependency graph for FctERR.c:



Functions

- FctERR HALERRtoFCTERR (HAL_StatusTypeDef status)

    Convert HAL_StatusTypeDef to FctERR.

4.3.1 Detailed Description

errors to SMFSW FctERR code

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.3.2 Function Documentation

4.3.2.1 HALERRtoFCTERR()

```
FctERR HALERRtoFCTERR (
             HAL_StatusTypeDef status )
```

Convert HAL_StatusTypeDef to FctERR.

Parameters

`in`	*status*	- HAL_StatusTypeDef status

Returns

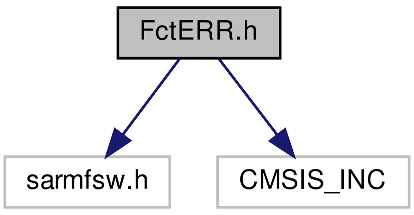
FctERR status

4.4 FctERR.h File Reference

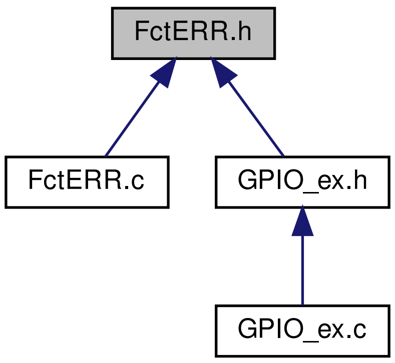
errors to SMFSW FctERR declarations

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

Include dependency graph for FctERR.h:



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



Typedefs

- typedef enum FctERR FctERR

Enumerations

- enum FctERR {
ERR_OK = 0, ERR_SPEED = -1, ERR_RANGE = -2, ERR_TIMEOUT = -3,
ERR_VALUE = -4, ERR_OVERFLOW = -5, ERR_MATH = -6, ERR_ENABLED = -7,
ERR_DISABLED = -8, ERR_BUSY = -9, ERR_NOTAVAIL = -10, ERR_RXEMPTY = -11,
ERR_TXFULL = -12, ERR_BUSOFF = -13, ERR_OVERRUN = -14, ERR_FRAMING = -15,
ERR_PARITY = -16, ERR_NOISE = -17, ERR_IDLE = -18, ERR_FAULT = -19,
ERR_BREAK = -20, ERR_CRC = -21, ERR_ARBITR = -22, ERR_PROTECT = -23,
ERR_UNDERFLOW = -24, ERR_UNDERRUN = -25, ERR_COMMON = -26, ERR_LINSYNC = -27,
ERR_FAILED = -28, ERR_QFULL = -29, ERR_CMD = -30, ERR_NOTIMPLEM = -31,
ERR_MEMORY = -32, ERR_INSTANCE = -33 }

    Enum of low/mid level functions return state.

Functions

- FctERR HALERRtoFCTERR (HAL_StatusTypeDef status)

    Convert HAL_StatusTypeDef to FctERR.

4.4.1 Detailed Description

errors to SMFSW FctERR declarations

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.4.2 Typedef Documentation

4.4.2.1 FctERR

```
typedef enum FctERR FctERR
```

4.4.3 Enumeration Type Documentation

4.4.3.1 FctERR

```
enum FctERR
```

Enum of low/mid level functions return state.

Note

TODO: Fix !defined lines when __mx_lwip_H set (should not cause any harm, but ugly and set to cause issues sometime, somehow)

Enumerator

ERR_OK	OK.
ERR_SPEED	This device does not work in the active speed mode.
ERR_RANGE	Parameter out of range.
ERR_TIMEOUT	Abort on timeout error.
ERR_VALUE	Parameter of incorrect value.
ERR_OVERFLOW	Overflow.
ERR_MATH	Overflow during evaluation.
ERR_ENABLED	Device is enabled.
ERR_DISABLED	Device is disabled.
ERR_BUSY	Device is busy.
ERR_NOTAVAIL	Requested value or method not available.
ERR_RXEMPTY	No data in receiver.
ERR_TXFULL	Transmitter is full.
ERR_BUSOFF	Bus not available.
ERR_OVERRUN	Overrun error is detected.
ERR_FRAMING	Framing error is detected.
ERR_PARITY	Parity error is detected.
ERR_NOISE	Noise error is detected.
ERR_IDLE	Idle error is detected.
ERR_FAULT	Fault error is detected.
ERR_BREAK	Break char is received during communication.
ERR_CRC	CRC error is detected.
ERR_ARBITR	A node lost arbitration. This error occurs if two nodes start transmission at the same time.
ERR_PROTECT	Protection error is detected.
ERR_UNDERFLOW	Underflow error is detected.
ERR_UNDERRUN	Underrun error is detected.
ERR_COMMON	Common error of a device.
ERR_LINSYNC	LIN synchronization error is detected.
ERR_FAILED	Requested functionality or process failed.
ERR_QFULL	Queue is full.
ERR_CMD	Command error is detected.
ERR_NOTIMPLEM	Function not implemented error.
ERR_MEMORY	Memory error.
ERR_INSTANCE	Instance error.

4.4.4 Function Documentation

4.4.4.1 HALERRtoFCTERR()

```
FctERR HALERRtoFCTERR (
            HAL_StatusTypeDef status )
```

Convert HAL_StatusTypeDef to FctERR.

Parameters

in	*status*	- HAL_StatusTypeDef status

Returns

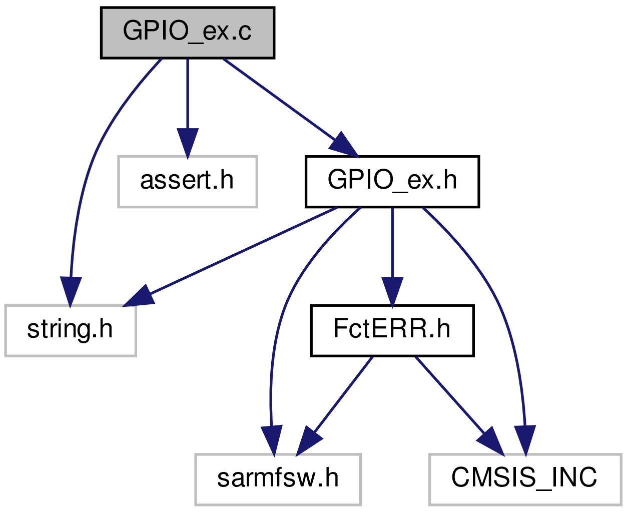
FctERR status

4.5 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.

4.5.1 Detailed Description

Simple extension for GPIOs.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.5.2 Macro Definition Documentation

4.5.2.1 MAX_PINS_PORT

```
#define MAX_PINS_PORT 16
```

4.5.3 Function Documentation

4.5.3.1 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

4.5.3.2 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

4.5.3.3 str_GPIO_name()

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

Get name from Port, Pin.

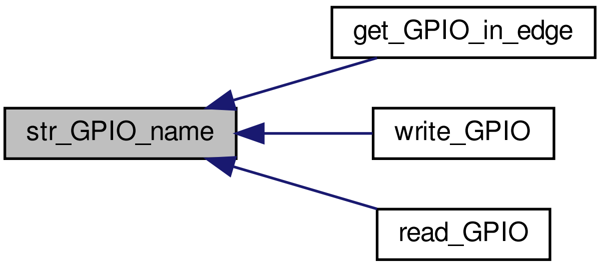
Parameters

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:

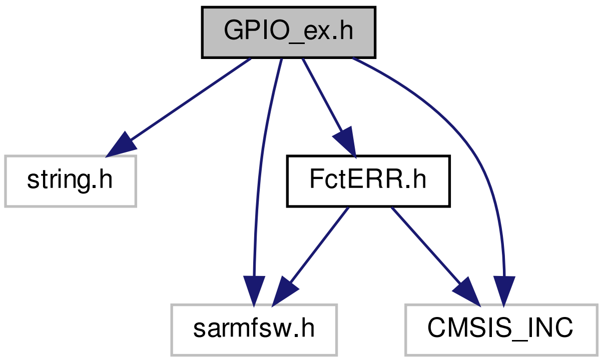


4.6 GPIO_ex.h File Reference

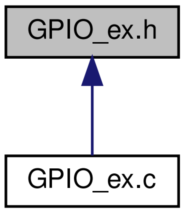
Simple extension for GPIOs.

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

4.6.1 Detailed Description

Simple extension for GPIOs.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.6.2 Typedef Documentation

4.6.2.1 eActOut

```
typedef enum ActOut eActOut
```

4.6.2.2 GPIO_in

```
typedef struct GPIO_in GPIO_in
```

4.6.3 Enumeration Type Documentation

4.6.3.1 ActOut

```
enum ActOut
```

Logic output possible actions enumeration.

Enumerator

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

4.6.4 Function Documentation

4.6.4.1 get_GPIO_in()

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

Get GPIO_in input value.

Parameters

in	*in*	- input instance

Returns

Input value

4.6.4.2 get_GPIO_in_edge()

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

Get GPIO_in input edge.

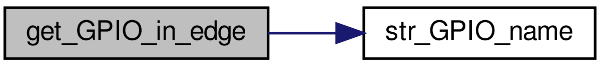
Parameters

in	*in*	- input instance

Returns

Input edge

Here is the call graph for this function:



4.6.4.3 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

4.6.4.4 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

4.6.4.5 read_GPIO()

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

Read GPIO.

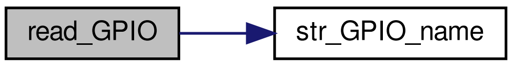
Parameters

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

Returns

Pin state

Here is the call graph for this function:



4.6.4.6 str_GPIO_name()

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

Get name from Port, Pin.

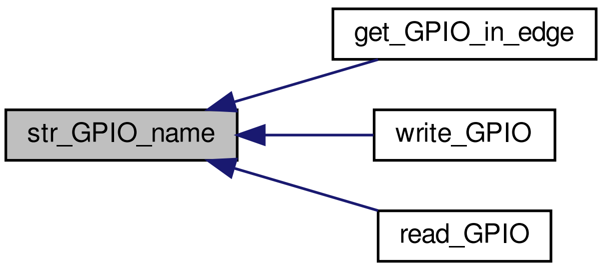
Parameters

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:



4.6.4.7 write_GPIO()

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

Write GPIO.

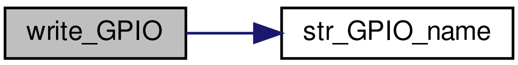
Parameters

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:

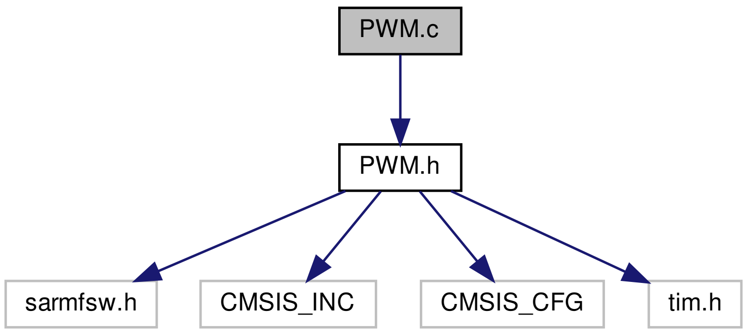


4.7 PWM.c File Reference

Straightforward PWM handling.

`#include "PWM.h"`

Include dependency graph for PWM.c:



Functions

- 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_TIM_Freq (TIM_HandleTypeDef ∗pTim, uint32_t freq)

  Set TIM module frequency.
- 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)

4.7.1 Detailed Description

Straightforward PWM handling.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.7.2 Function Documentation

4.7.2.1 init_PWM_Chan()

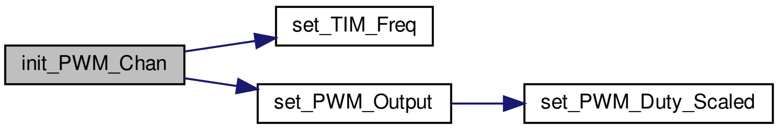
```
HAL_StatusTypeDef init_PWM_Chan (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t freq )
```

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:



4.7.2.2 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)

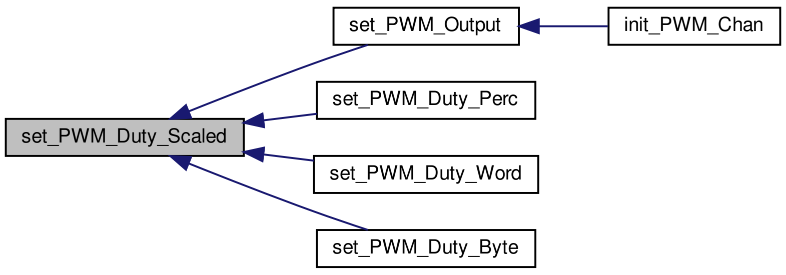
Parameters

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:



4.7.2.3 set_TIM_Freq()

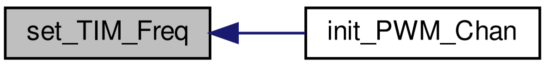
```
HAL_StatusTypeDef set_TIM_Freq (
            TIM_HandleTypeDef * pTim,
            uint32_t freq )
```

Set TIM module frequency.

Parameters

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

Here is the caller graph for this function:

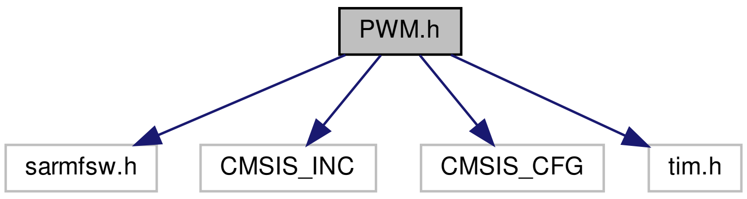


4.8 PWM.h File Reference

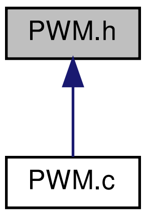
Straightforward PWM handling.

```
#include "sarmfsw.h"
#include <CMSIS_INC>
#include <CMSIS_CFG>
#include "tim.h"
```

Include dependency graph for PWM.h:



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



Functions

- 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_TIM_Freq (TIM_HandleTypeDef ∗pTim, uint32_t freq)

  Set TIM module frequency.
- 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)

4.8.1 Detailed Description

Straightforward PWM handling.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.8.2 Function Documentation

4.8.2.1 init_PWM_Chan()

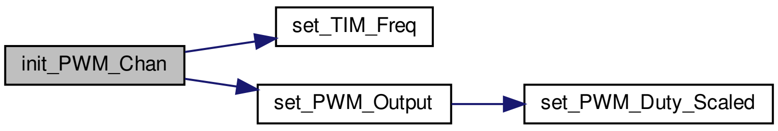
```
HAL_StatusTypeDef init_PWM_Chan (
            TIM_HandleTypeDef * pTim,
            uint32_t chan,
            uint16_t freq )
```

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:



4.8.2.2 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)

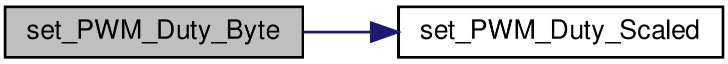
Parameters

`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:



4.8.2.3 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)

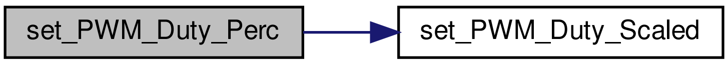
Parameters

`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:



4.8.2.4 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)

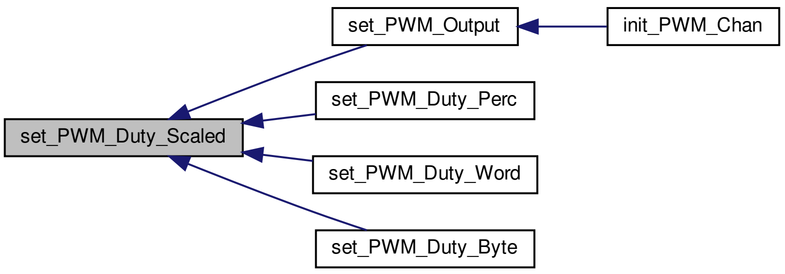
Parameters

`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:



4.8.2.5 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)

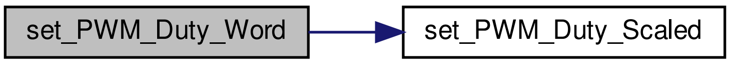
Parameters

`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:



4.8.2.6 set_PWM_Output()

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

Set PWM channel output on/off.

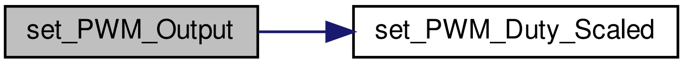
Parameters

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

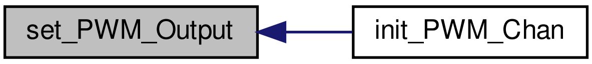
Returns

HAL Status

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.7 set_TIM_Freq()

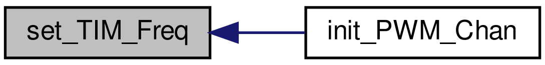
```
HAL_StatusTypeDef set_TIM_Freq (
            TIM_HandleTypeDef * pTim,
            uint32_t freq )
```

Set TIM module frequency.

Parameters

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

Here is the caller graph for this function:



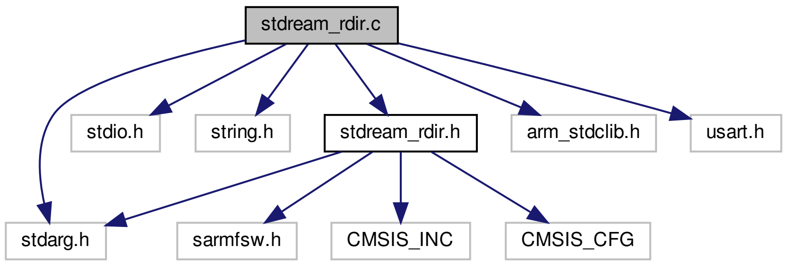
4.9 stdream_rdir.c File Reference

Stream redirection.

```
#include <stdarg.h>
#include <stdio.h>
#include <string.h>
#include "stdream_rdir.h"
#include "arm_stdclib.h"
```

```
#include "usart.h"
```

Include dependency graph for stdream_rdir.c:



Functions

- void print_itm_port (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)
- int32_t get_fp_dec (float f, uint8_t nb)

  Get floating point number decimal part.

4.9.1 Detailed Description

Stream redirection.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.9.2 Function Documentation

4.9.2.1 get_fp_dec()

```
int32_t get_fp_dec (
            float f,
            uint8_t nb )
```

Get floating point number decimal part.

Note

in need to print floats, add '-u _printf_float' in Linker options

Warning

enabling floating point support from linker seems to fubar printing long variables

Parameters

in	*f*	- floating point value
in	*nb*	- Number of decimal to get after floating point

Returns

nb decimal part as integer

4.9.2.2 print_itm_port()

```
void print_itm_port (
            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

4.9.2.3 printf_ITM()

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

4.9.2.4 printf_rdir()

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

4.9.2.5 vprintf_ITM()

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

4.9.2.6 vprintf_rdir()

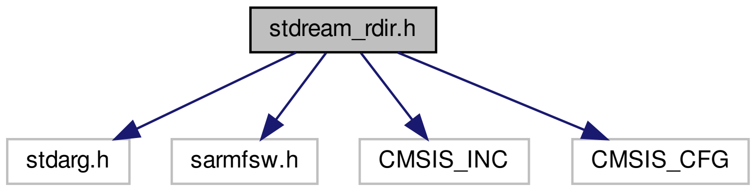
```
int vprintf_rdir (
             char * str,
             va_list args )
```

4.10 stdream_rdir.h File Reference

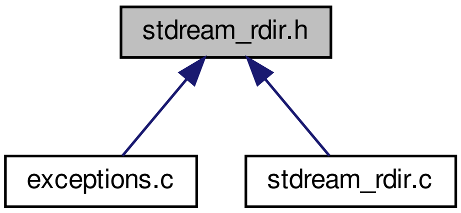
Stream redirection header.

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

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.

Functions

- void print_itm_port (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)
- int32_t get_fp_dec (float f, uint8_t nb)

  Get floating point number decimal part.

4.10.1 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, stings will be printed to ITM0 port only

4.10.2 Macro Definition Documentation

4.10.2.1 printf

```
#define printf printf_rdir
```

Shadowing printf.

4.10.2.2 vprintf

```
#define vprintf vprintf_rdir
```

Shadowing vprintf.

4.10.3 Function Documentation

4.10.3.1 get_fp_dec()

```
int32_t get_fp_dec (
            float f,
            uint8_t nb )
```

Get floating point number decimal part.

Note

in need to print floats, add '-u _printf_float' in Linker options

Warning

enabling floating point support from linker seems to fubar printing long variables

Parameters

in	*f*	- floating point value
in	*nb*	- Number of decimal to get after floating point

Returns

nb decimal part as integer

4.10.3.2 print_itm_port()

```
void print_itm_port (
             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

4.10.3.3 printf_ITM()

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

4.10.3.4 printf_rdir()

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

4.10.3.5 vprintf_ITM()

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

4.10.3.6 vprintf_rdir()

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

Index