

HARMcksL: ARM HAL toolbox (yet STM32 oriented)

0.5

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	1
2.1	File List	1
3	Class Documentation	2
3.1	GPIO_in Struct Reference	2
3.1.1	Detailed Description	3
3.1.2	Member Data Documentation	3
4	File Documentation	4
4.1	exceptions.c File Reference	4
4.1.1	Detailed Description	4
4.1.2	Function Documentation	5
4.2	exceptions.h File Reference	6
4.2.1	Detailed Description	7
4.2.2	Macro Definition Documentation	7
4.2.3	Function Documentation	8
4.3	FctERR.c File Reference	8
4.3.1	Detailed Description	9
4.3.2	Function Documentation	9
4.4	FctERR.h File Reference	9
4.4.1	Detailed Description	11
4.4.2	Typedef Documentation	11
4.4.3	Enumeration Type Documentation	11
4.4.4	Function Documentation	12
4.5	GPIO_ex.c File Reference	12
4.5.1	Detailed Description	13
4.5.2	Macro Definition Documentation	14

4.5.3	Function Documentation	14
4.6	GPIO_ex.h File Reference	15
4.6.1	Detailed Description	17
4.6.2	Typedef Documentation	17
4.6.3	Enumeration Type Documentation	17
4.6.4	Function Documentation	17
4.7	PWM.c File Reference	20
4.7.1	Detailed Description	21
4.7.2	Function Documentation	21
4.8	PWM.h File Reference	23
4.8.1	Detailed Description	24
4.8.2	Function Documentation	24
4.9	stdream_rdir.c File Reference	27
4.9.1	Detailed Description	28
4.9.2	Function Documentation	28
4.10	stdream_rdir.h File Reference	29
4.10.1	Detailed Description	30
4.10.2	Macro Definition Documentation	31
4.10.3	Function Documentation	31
Index		33

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:

exceptions.c	
Debug tool helpers functions	4
exceptions.h	
Debug tool and helpers declaration	6
FctERR.c	
Errors to SMFSW FctERR code	8
FctERR.h	
Errors to SMFSW FctERR declarations	9
GPIO_ex.c	
Simple extension for GPIOs	12
GPIO_ex.h	
Simple extension for GPIOs	15
PWM.c	
Simple PWM handling	20
PWM.h	
Simple PWM handling	23
stdream_rdir.c	
Stream redirection	27
stdream_rdir.h	
Stream redirection header	29

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 hln](#)
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 struct { ... } GPIO_in::cfg

3.1.2.2 bool GPIO_in::done

State change done.

3.1.2.3 eEdge GPIO_in::edge

Input edge.

3.1.2.4 uint16_t GPIO_in::filt

Filter time (ms)

3.1.2.5 uint16_t GPIO_in::GPIO_Pin

HAL GPIO pin.

3.1.2.6 GPIO_TypeDef* GPIO_in::GPIOx

HAL GPIO instance.

3.1.2.7 uint32_t GPIO_in::hln

Filter time.

3.1.2.8 bool GPIO_in::in

Input value.

3.1.2.9 bool GPIO_in::mem

Memo value.

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

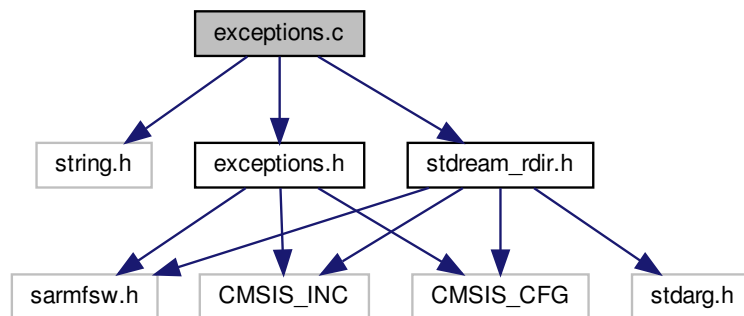
- [GPIO_ex.h](#)

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[])
- void [HardFault_Handler_callback](#) (uint32_t stack[])
- void [Error_Handler_callback](#) (uint32_t stack[])

4.1.1 Detailed Description

Debug tool helpers functions.

Author

SMFSW

Version

v0.5

Date

2017

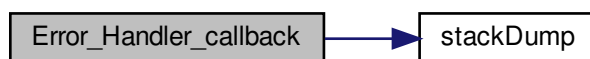
Copyright

MIT (c) 2017, SMFSW

4.1.2 Function Documentation

4.1.2.1 void Error_Handler_callback (uint32_t *stack*[])

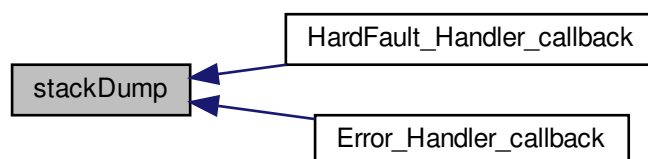
Here is the call graph for this function:

4.1.2.2 void HardFault_Handler_callback (uint32_t *stack*[])

Here is the call graph for this function:

4.1.2.3 void stackDump (uint32_t *stack*[])

Here is the caller graph for this function:

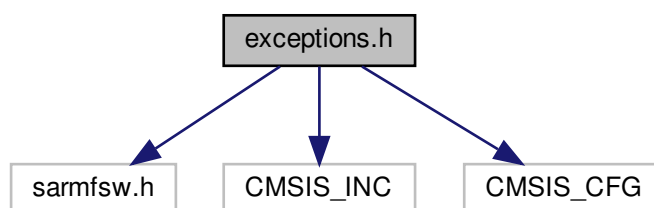


4.2 exceptions.h File Reference

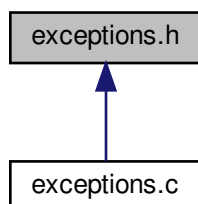
Debug tool and helpers declaration.

```
#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)`
*The exception_Handler should be called with corresponding exception name **e** as parameter.*
- `#define dump_stack()`

Functions

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

4.2.1 Detailed Description

Debug tool and helpers declaration.

Author

SMFSW

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.2.2 Macro Definition Documentation

4.2.2.1 #define dump_stack()

Value:

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

4.2.2.2 #define exception_Handler(e)

Value:

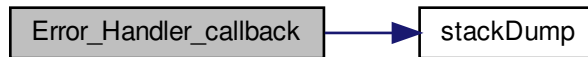
```
__asm(  "tst lr, #4 \n"      \
        "ite EQ \n"        \
        "mrseq r0, MSP \n"  \
        "mrsne r0, PSP \n"  \
        "b " #e "_Handler_callback \n")
```

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

4.2.3 Function Documentation

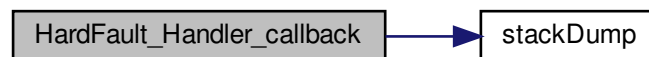
4.2.3.1 void Error_Handler_callback (uint32_t *stack*[])

Here is the call graph for this function:



4.2.3.2 void HardFault_Handler_callback (uint32_t *stack*[])

Here is the call graph for this function:

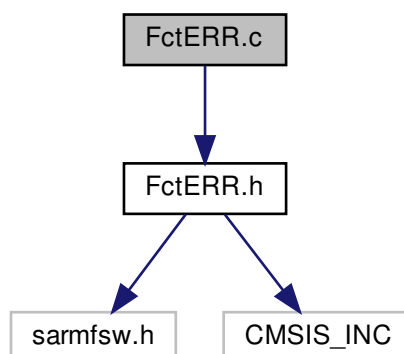


4.3 FctERR.c File Reference

errors to SMFSW FctERR code

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

Include dependency graph for FctERR.c:



Functions

- [FctERR HALERRtoFCTERR](#) (HAL_StatusTypeDef st)
Convert HAL_StatusTypeDef to FctERR.

4.3.1 Detailed Description

errors to SMFSW FctERR code

Author

SMFSW

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.3.2 Function Documentation

4.3.2.1 FctERR HALERRtoFCTERR (HAL_StatusTypeDef st)

Convert HAL_StatusTypeDef to FctERR.

Parameters

in	st	- HAL_StatusTypeDef status
----	----	----------------------------

Returns

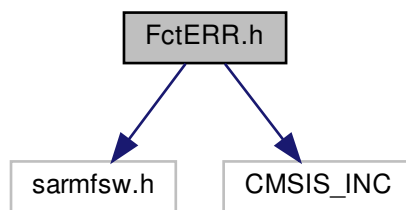
FctERR status

4.4 FctERR.h File Reference

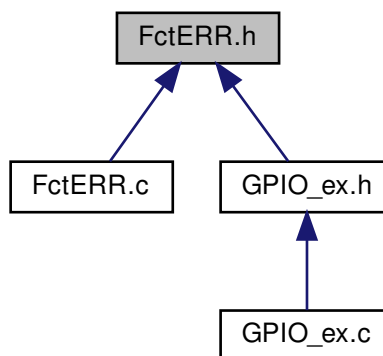
errors to SMFSW FctERR declarations

```
#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 [EnumFctERR](#) FctERR

Enumerations

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

Enum of high level functions return state.

Functions

- [FctERR HALERRtoFCTERR](#) (HAL_StatusTypeDef st)
Convert HAL_StatusTypeDef to FctERR.

4.4.1 Detailed Description

errors to SMFSW FctERR declarations

Author

SMFSW

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.4.2 Typedef Documentation

4.4.2.1 typedef enum EnumFctERR FctERR

4.4.3 Enumeration Type Documentation

4.4.3.1 enum EnumFctERR

Enum of high level functions return state.

Enumerator

ERR_OK OK.

ERR_SPEED This device does not work in the active speed mode.

ERR_RANGE Parameter out of range.

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_ARBTR 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_TIMEOUT Abort on timeout error.

ERR_NOTIMPLEM Function not implemented error.

ERR_MEMORY Memory error.

ERR_INSTANCE Instance error.

4.4.4 Function Documentation

4.4.4.1 FctERR HALERRtoFCTERR (HAL_StatusTypeDef st)

Convert HAL_StatusTypeDef to FctERR.

Parameters

in	st	- HAL_StatusTypeDef status
----	----	----------------------------

Returns

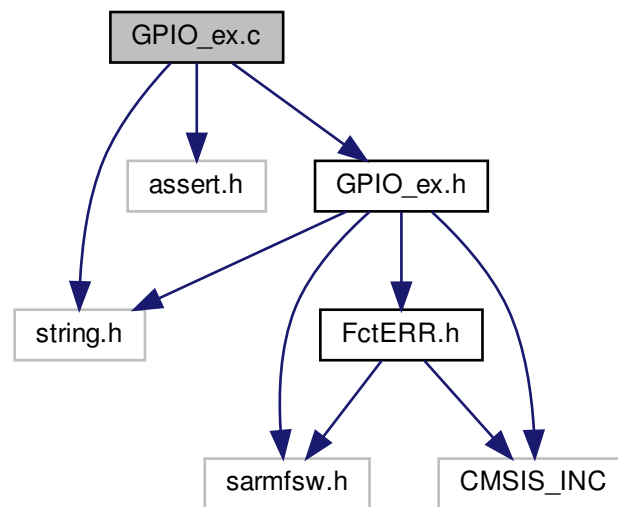
FctERR status

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

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.5.2 Macro Definition Documentation

4.5.2.1 #define MAX_PINS_PORT 16

4.5.3 Function Documentation

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

Get name from Port, Pin.

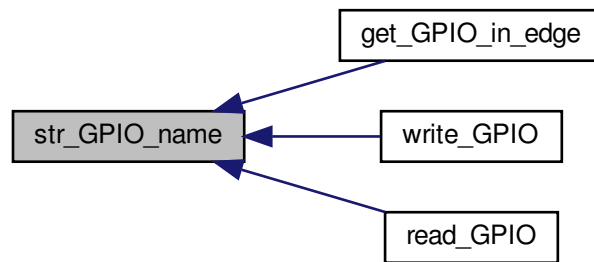
Parameters

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

Returns

Error code

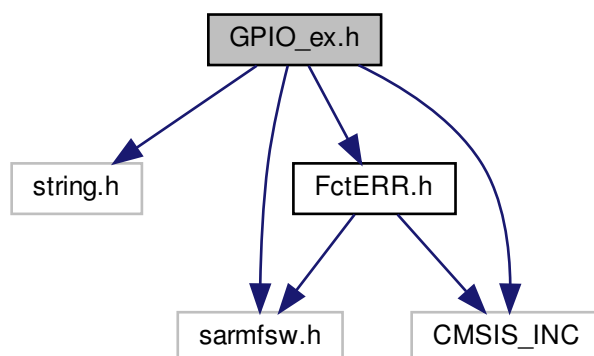
Here is the caller graph for this function:



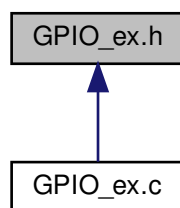
4.6 GPIO_ex.h File Reference

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

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.6.2 Typedef Documentation

4.6.2.1 typedef enum ActOut eActOut

4.6.2.2 typedef struct GPIO_in GPIO_in

4.6.3 Enumeration Type Documentation

4.6.3.1 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 bool get_GPIO_in (GPIO_in * in)

Get [GPIO_in](#) input value.

Parameters

in	in	- input instance
----	----	------------------

Returns

Input value

4.6.4.2 bool get_GPIO_in_edge (GPIO_in * in)

Get [GPIO_in](#) input edge.

Parameters

in	<i>in</i>	- input instance
----	-----------	------------------

Returns

Input edge

Here is the call graph for this function:

**4.6.4.3 void GPIO_in_handler (GPIO_in * in)**

Handles [GPIO_in](#) read and treatment.

Parameters

in, out	<i>in</i>	- input instance to handle
---------	-----------	----------------------------

Returns

Nothing

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

Initialize [GPIO_in](#) instance.

Parameters

in, out	<i>in</i>	- input instance to initialize
in	<i>GPIOx</i>	- port to write to
in	<i>GPIO_Pin</i>	- pin to write to
in	<i>filter</i>	- input filtering time

Returns

Nothing

4.6.4.5 GPIO_PinState read_GPIO (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)

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

Get name from Port, Pin.

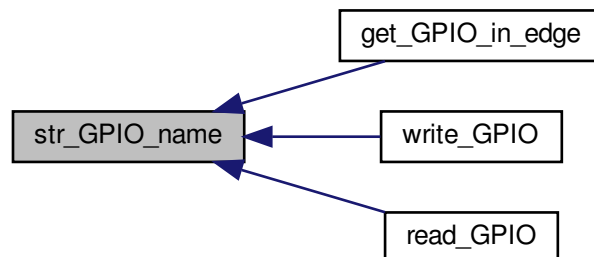
Parameters

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 void write_GPIO (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, eActOut Act)

Write GPIO.

Parameters

in	<i>GPIOx</i>	- port to write to
in	<i>GPIO_Pin</i>	- pin to write to
in	<i>Act</i>	- type of write

Returns

Nothing

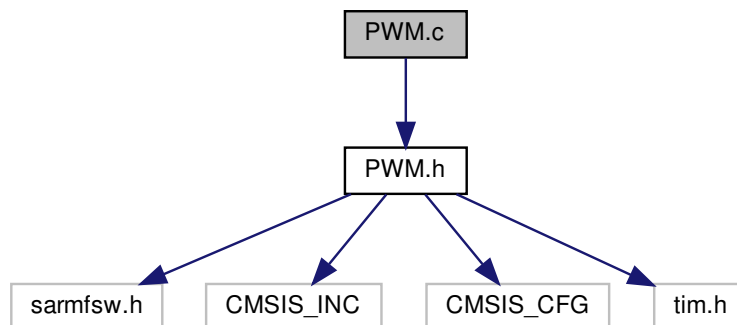
Here is the call graph for this function:

**4.7 PWM.c File Reference**

Simple PWM handling.

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

Include dependency graph for PWM.c:



Functions

- HAL_StatusTypeDef [set_PWM_Freq](#) (TIM_HandleTypeDef *pTim, uint32_t freq)
Set TIM module PWM frequency for channel.
- HAL_StatusTypeDef [write_CCR](#) (TIM_HandleTypeDef *pTim, uint32_t chan, uint16_t CCR_val)
Low level TIM module PWM duty cycle write.
- 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

Simple PWM handling.

Author

SMFSW

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.7.2 Function Documentation

4.7.2.1 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)

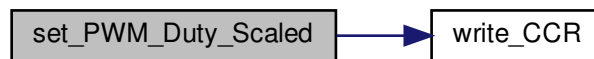
Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>chan</i>	- Channel to write
in	<i>duty</i>	- Scaled duty cycle value to write
in	<i>scale</i>	- Full scale value

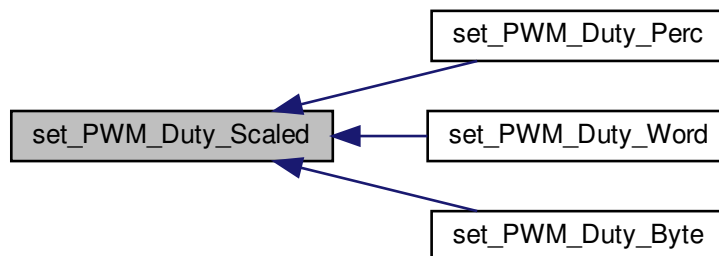
Returns

HAL Status

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.2 HAL_StatusTypeDef set_PWM_Freq (TIM_HandleTypeDef * pTim, uint32_t freq)

Set TIM module PWM frequency for channel.

Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>freq</i>	- Desired PWM frequency

4.7.2.3 HAL_StatusTypeDef write_CCR (TIM_HandleTypeDef * *pTim*, uint32_t *chan*, uint16_t *CCR_val*)

Low level TIM module PWM duty cycle write.

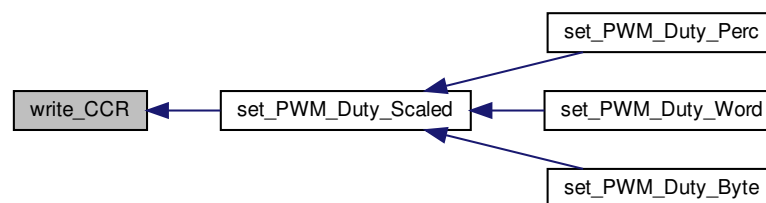
Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>chan</i>	- Channel to write
in	<i>CCR_val</i>	- Scaled duty cycle for CCR register

Returns

HAL Status

Here is the caller graph for this function:

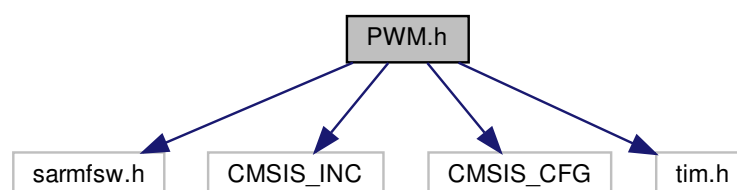


4.8 PWM.h File Reference

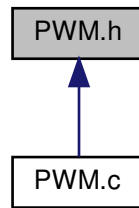
Simple PWM handling.

```
#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 [set_PWM_Freq](#) (TIM_HandleTypeDef *pTim, uint32_t freq)
Set TIM module PWM frequency for 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)
- 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

Simple PWM handling.

Author

SMFSW

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.8.2 Function Documentation

4.8.2.1 HAL_StatusTypeDef set_PWM_Duty_Byte (TIM_HandleTypeDef * pTim, uint32_t chan, uint8_t duty)

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

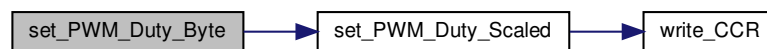
Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>chan</i>	- Channel to write
in	<i>duty</i>	- Scaled duty cycle value to write

Returns

HAL Status

Here is the call graph for this function:



4.8.2.2 HAL_StatusTypeDef set_PWM_Duty_Perc (TIM_HandleTypeDef * *pTim*, uint32_t *chan*, uint16_t *duty*)

Set TIM module PWM duty cycle (percents)

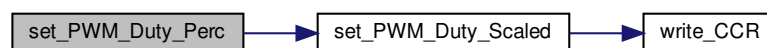
Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>chan</i>	- Channel to write
in	<i>duty</i>	- Scaled duty cycle value to write

Returns

HAL Status

Here is the call graph for this function:



4.8.2.3 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)

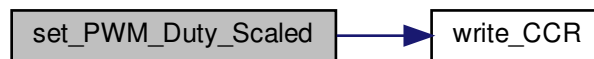
Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>chan</i>	- Channel to write
in	<i>duty</i>	- Scaled duty cycle value to write
in	<i>scale</i>	- Full scale value

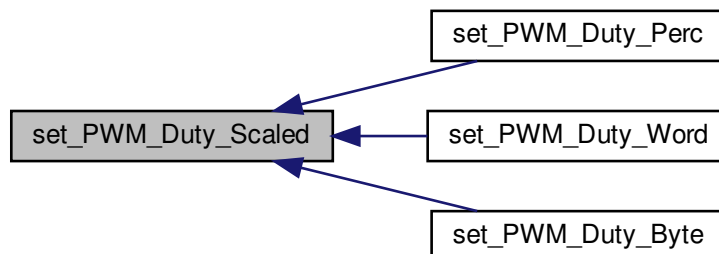
Returns

HAL Status

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.4 HAL_StatusTypeDef set_PWM_Duty_Word (TIM_HandleTypeDef * pTim, uint32_t chan, uint16_t duty)

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

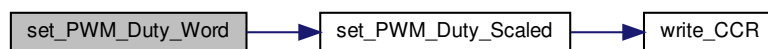
Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>chan</i>	- Channel to write
in	<i>duty</i>	- Scaled duty cycle value to write

Returns

HAL Status

Here is the call graph for this function:



4.8.2.5 HAL_StatusTypeDef set_PWM_Freq (TIM_HandleTypeDef * *pTim*, uint32_t *freq*)

Set TIM module PWM frequency for channel.

Parameters

in, out	<i>pTim</i>	- pointer to TIM instance for PWM generation
in	<i>freq</i>	- Desired PWM frequency

4.9 stdream_rdir.c File Reference

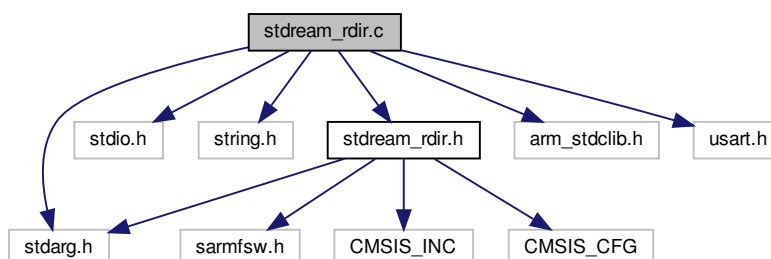
Stream redirection.

```

#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_uart` (char *ptr, int len)
- void `print_itm_port` (int port, const char *msg, int len)
- int `printf_ITM` (char *string,...)
- int `vprintf_ITM` (char *string, va_list args)
- int `printf_rdir` (char *string,...)
- int `vprintf_rdir` (char *string, va_list args)

4.9.1 Detailed Description

Stream redirection.

Author

SMFSW

Version

v0.5

Date

2017

Copyright

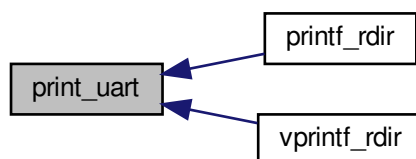
MIT (c) 2017, SMFSW

4.9.2 Function Documentation

4.9.2.1 void `print_itm_port` (int *port*, const char * *msg*, int *len*)

4.9.2.2 void `print_uart` (char * *ptr*, int *len*)

Here is the caller graph for this function:



4.9.2.3 int printf_ITM (char * *string*, ...)

4.9.2.4 int printf_rdir (char * *string*, ...)

Here is the call graph for this function:



4.9.2.5 int vprintf_ITM (char * *string*, va_list *args*)

4.9.2.6 int vprintf_rdir (char * *string*, va_list *args*)

Here is the call graph for this function:

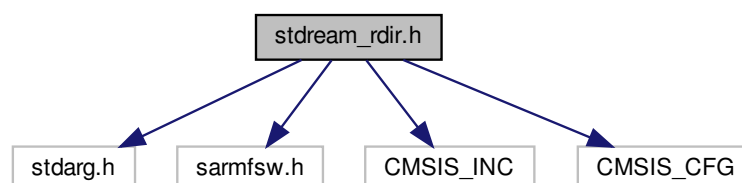


4.10 stdream_rdir.h File Reference

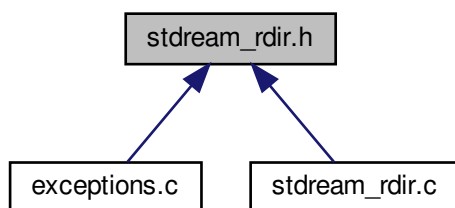
Stream redirection header.

```
#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 use.
- `#define vprintf vprintf_rdir`
Shadowing vprintf use.

Functions

- void `print_itm_port` (int port, const char *msg, int len)
- int `printf_ITM` (char *string,...)
- int `vprintf_ITM` (char *string, va_list args)
- int `printf_rdir` (char *string,...)
- int `vprintf_rdir` (char *string, va_list args)

4.10.1 Detailed Description

Stream redirection header.

Author

SMFSW

Version

v0.5

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.10.2 Macro Definition Documentation

4.10.2.1 #define printf printf_rdir

Shadowing printf use.

4.10.2.2 #define vprintf vprintf_rdir

Shadowing vprintf use.

4.10.3 Function Documentation

4.10.3.1 void print_itm_port (int *port*, const char * *msg*, int *len*)

4.10.3.2 int printf_ITM (char * *string*, ...)

4.10.3.3 int printf_rdir (char * *string*, ...)

Here is the call graph for this function:



4.10.3.4 int vprintf_ITM (char * *string*, va_list *args*)

4.10.3.5 int vprintf_rdir (char * *string*, va_list *args*)

Here is the call graph for this function:



Index

ActOut
 GPIO_ex.h, 17

cfg
 GPIO_in, 3

done
 GPIO_in, 3

dump_stack
 exceptions.h, 7

eActOut
 GPIO_ex.h, 17

ERR_ARBITR
 FctERR.h, 12

ERR_BREAK
 FctERR.h, 12

ERR_BUSOFF
 FctERR.h, 12

ERR_BUSY
 FctERR.h, 11

ERR_CMD
 FctERR.h, 12

ERR_COMMON
 FctERR.h, 12

ERR_CRC
 FctERR.h, 12

ERR_DISABLED
 FctERR.h, 11

ERR_ENABLED
 FctERR.h, 11

ERR_FAILED
 FctERR.h, 12

ERR_FAULT
 FctERR.h, 12

ERR_FRAMING
 FctERR.h, 12

ERR_IDLE
 FctERR.h, 12

ERR_INSTANCE
 FctERR.h, 12

ERR_LINSYNC
 FctERR.h, 12

ERR_MATH
 FctERR.h, 11

ERR_MEMORY
 FctERR.h, 12

ERR_NOISE
 FctERR.h, 12

ERR_NOTAVAIL
 FctERR.h, 11

ERR_NOTIMPLEM
 FctERR.h, 12

ERR_OVERFLOW
 FctERR.h, 11

ERR_OVERRUN
 FctERR.h, 12

ERR_OK
 FctERR.h, 11

ERR_PARITY
 FctERR.h, 12

ERR_PROTECT
 FctERR.h, 12

ERR_QFULL
 FctERR.h, 12

ERR_RANGE
 FctERR.h, 11

ERR_RXEMPTY
 FctERR.h, 11

ERR_SPEED
 FctERR.h, 11

ERR_TIMEOUT
 FctERR.h, 12

ERR_TXFULL
 FctERR.h, 11

ERR_UNDERFLOW
 FctERR.h, 12

ERR_UNDERRUN
 FctERR.h, 12

ERR_VALUE
 FctERR.h, 11

edge
 GPIO_in, 3

EnumFctERR
 FctERR.h, 11

Error_Handler_callback
 exceptions.c, 5
 exceptions.h, 8

exception_Handler
 exceptions.h, 7

exceptions.c, 4
 Error_Handler_callback, 5
 HardFault_Handler_callback, 5
 stackDump, 5

exceptions.h, 6
 dump_stack, 7
 Error_Handler_callback, 8
 exception_Handler, 7
 HardFault_Handler_callback, 8

FctERR.c, 8
 HALERRtoFCTERR, 9

FctERR.h, 9
 ERR_ARBITR, 12
 ERR_BREAK, 12
 ERR_BUSOFF, 12
 ERR_BUSY, 11
 ERR_CMD, 12
 ERR_COMMON, 12
 ERR_CRC, 12
 ERR_DISABLED, 11

- ERR_ENABLED, 11
- ERR_FAILED, 12
- ERR_FAULT, 12
- ERR_FRAMING, 12
- ERR_IDLE, 12
- ERR_INSTANCE, 12
- ERR_LINSYNC, 12
- ERR_MATH, 11
- ERR_MEMORY, 12
- ERR_NOISE, 12
- ERR_NOTAVAIL, 11
- ERR_NOTIMPLEM, 12
- ERR_OVERFLOW, 11
- ERR_OVERRUN, 12
- ERR_OK, 11
- ERR_PARITY, 12
- ERR_PROTECT, 12
- ERR_QFULL, 12
- ERR_RANGE, 11
- ERR_RXEMPTY, 11
- ERR_SPEED, 11
- ERR_TIMEOUT, 12
- ERR_TXFULL, 11
- ERR_UNDERFLOW, 12
- ERR_UNDERRUN, 12
- ERR_VALUE, 11
- EnumFctERR, 11
- FctERR, 11
- HALERRtoFCTERR, 12
- FctERR
 - FctERR.h, 11
- filt
 - GPIO_in, 3
- GPIO_Pin
 - GPIO_in, 3
- GPIO_ex.c, 12
 - GPIO_in_handler, 14
 - GPIO_in_init, 14
 - MAX_PINS_PORT, 14
 - str_GPIO_name, 14
- GPIO_ex.h, 15
 - ActOut, 17
 - eActOut, 17
 - GPIO_in, 17
 - GPIO_in_handler, 18
 - GPIO_in_init, 18
 - get_GPIO_in, 17
 - get_GPIO_in_edge, 18
 - read_GPIO, 19
 - Reset, 17
 - Set, 17
 - str_GPIO_name, 19
 - Toggle, 17
 - write_GPIO, 20
- GPIO_in, 2
 - cfg, 3
 - done, 3
 - edge, 3
 - filt, 3
 - GPIO_Pin, 3
 - GPIO_ex.h, 17
 - GPIOx, 3
 - hIn, 3
 - in, 3
 - mem, 3
 - GPIO_in_handler
 - GPIO_ex.c, 14
 - GPIO_ex.h, 18
 - GPIO_in_init
 - GPIO_ex.c, 14
 - GPIO_ex.h, 18
 - GPIOx
 - GPIO_in, 3
 - get_GPIO_in
 - GPIO_ex.h, 17
 - get_GPIO_in_edge
 - GPIO_ex.h, 18
- HALERRtoFCTERR
 - FctERR.c, 9
 - FctERR.h, 12
- hIn
 - GPIO_in, 3
- HardFault_Handler_callback
 - exceptions.c, 5
 - exceptions.h, 8
- in
 - GPIO_in, 3
- MAX_PINS_PORT
 - GPIO_ex.c, 14
- mem
 - GPIO_in, 3
- PWM.c, 20
 - set_PWM_Duty_Scaled, 21
 - set_PWM_Freq, 22
 - write_CCR, 22
- PWM.h, 23
 - set_PWM_Duty_Byte, 24
 - set_PWM_Duty_Perc, 25
 - set_PWM_Duty_Scaled, 25
 - set_PWM_Duty_Word, 26
 - set_PWM_Freq, 27
- print_itm_port
 - stdream_rdir.c, 28
 - stdream_rdir.h, 31
- print_uart
 - stdream_rdir.c, 28
- printf
 - stdream_rdir.h, 31
- printf_ITM
 - stdream_rdir.c, 28
 - stdream_rdir.h, 31
- printf_rdir
 - stdream_rdir.c, 29

- stdream_rdir.h, 31
- read_GPIO
 - GPIO_ex.h, 19
- Reset
 - GPIO_ex.h, 17
- Set
 - GPIO_ex.h, 17
- set_PWM_Duty_Byte
 - PWM.h, 24
- set_PWM_Duty_Perc
 - PWM.h, 25
- set_PWM_Duty_Scaled
 - PWM.c, 21
 - PWM.h, 25
- set_PWM_Duty_Word
 - PWM.h, 26
- set_PWM_Freq
 - PWM.c, 22
 - PWM.h, 27
- stackDump
 - exceptions.c, 5
- stdream_rdir.c, 27
 - print_itm_port, 28
 - print_uart, 28
 - printf_ITM, 28
 - printf_rdir, 29
 - vprintf_ITM, 29
 - vprintf_rdir, 29
- stdream_rdir.h, 29
 - print_itm_port, 31
 - printf, 31
 - printf_ITM, 31
 - printf_rdir, 31
 - vprintf, 31
 - vprintf_ITM, 31
 - vprintf_rdir, 31
- str_GPIO_name
 - GPIO_ex.c, 14
 - GPIO_ex.h, 19
- Toggle
 - GPIO_ex.h, 17
- vprintf
 - stdream_rdir.h, 31
- vprintf_ITM
 - stdream_rdir.c, 29
 - stdream_rdir.h, 31
- vprintf_rdir
 - stdream_rdir.c, 29
 - stdream_rdir.h, 31
- write_CCR
 - PWM.c, 22
- write_GPIO
 - GPIO_ex.h, 20