sarmfsw: SMFSW Toolbox (for ARM, STM32)

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:

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

3 Class Documentation

3.1 StructBitfield16 Struct Reference

Bitfield 16b.

```
#include <arm_typedefs.h>
```

Public Attributes

- WORD b0:1

  Bit 0 (LSB)
- WORD b1:1

  Bit 1.
- WORD b2:1

  Bit 2.
- WORD b3:1

  Bit 3.
- WORD b4:1

  Bit 4.
- WORD b5:1

  Bit 5.
- WORD b6:1

  Bit 6.
- WORD b7:1

  Bit 7.
- WORD b8:1

  Bit 8.
- WORD b9:1

  Bit 9.
- WORD b10:1

  Bit 10.
- WORD b11:1

  Bit 11.
- WORD b12:1

  Bit 12.
- WORD b13:1

  Bit 13.
- WORD b14:1

  Bit 14.
- WORD b15:1

  Bit 15 (MSB)

3.1.1 Detailed Description

Bitfield 16b.

3.1.2 Member Data Documentation

3.1.2.1 b0

`WORD StructBitfield16::b0`

Bit 0 (LSB)

3.1.2.2 b1

`WORD StructBitfield16::b1`

Bit 1.

3.1.2.3 b10

`WORD StructBitfield16::b10`

Bit 10.

3.1.2.4 b11

`WORD StructBitfield16::b11`

Bit 11.

3.1.2.5 b12

`WORD StructBitfield16::b12`

Bit 12.

3.1.2.6 b13

`WORD StructBitfield16::b13`

Bit 13.

3.1.2.7 b14

```
WORD StructBitfield16::b14
```

Bit 14.

3.1.2.8 b15

```
WORD StructBitfield16::b15
```

Bit 15 (MSB)

3.1.2.9 b2

```
WORD StructBitfield16::b2
```

Bit 2.

3.1.2.10 b3

```
WORD StructBitfield16::b3
```

Bit 3.

3.1.2.11 b4

```
WORD StructBitfield16::b4
```

Bit 4.

3.1.2.12 b5

```
WORD StructBitfield16::b5
```

Bit 5.

3.1.2.13 b6

```
WORD StructBitfield16::b6
```

Bit 6.

3.1.2.14 b7

```
WORD StructBitfield16::b7
```

Bit 7.

3.1.2.15 b8

```
WORD StructBitfield16::b8
```

Bit 8.

3.1.2.16 b9

```
WORD StructBitfield16::b9
```

Bit 9.

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

- arm_typedefs.h

3.2 StructBitfield32 Struct Reference

Bitfield 32b.

```
#include <arm_typedefs.h>
```

Public Attributes

- DWORD b0:1

  Bit 0 (LSB)
- DWORD b1:1

  Bit 1.
- DWORD b2:1

  Bit 2.
- DWORD b3:1

  Bit 3.
- DWORD b4:1

  Bit 4.
- DWORD b5:1

  Bit 5.
- DWORD b6:1

  Bit 6.
- DWORD b7:1

  Bit 7.
- DWORD b8:1

Bit 8.

- DWORD b9:1

  Bit 9.
- DWORD b10:1

  Bit 10.
- DWORD b11:1

  Bit 11.
- DWORD b12:1

  Bit 12.
- DWORD b13:1

  Bit 13.
- DWORD b14:1

  Bit 14.
- DWORD b15:1

  Bit 15.
- DWORD b16:1

  Bit 16.
- DWORD b17:1

  Bit 17.
- DWORD b18:1

  Bit 18.
- DWORD b19:1

  Bit 19.
- DWORD b20:1

  Bit 20.
- DWORD b21:1

  Bit 21.
- DWORD b22:1

  Bit 22.
- DWORD b23:1

  Bit 23.
- DWORD b24:1

  Bit 24.
- DWORD b25:1

  Bit 25.
- DWORD b26:1

  Bit 26.
- DWORD b27:1

  Bit 27.
- DWORD b28:1

  Bit 28.
- DWORD b29:1

  Bit 29.
- DWORD b30:1

  Bit 30.
- DWORD b31:1

  Bit 31 (MSB)

3.2.1 Detailed Description

Bitfield 32b.

3.2.2 Member Data Documentation

3.2.2.1 b0

`DWORD StructBitfield32::b0`

Bit 0 (LSB)

3.2.2.2 b1

`DWORD StructBitfield32::b1`

Bit 1.

3.2.2.3 b10

`DWORD StructBitfield32::b10`

Bit 10.

3.2.2.4 b11

`DWORD StructBitfield32::b11`

Bit 11.

3.2.2.5 b12

`DWORD StructBitfield32::b12`

Bit 12.

3.2.2.6 b13

`DWORD StructBitfield32::b13`

Bit 13.

3.2.2.7 b14

```
DWORD StructBitfield32::b14
```

Bit 14.

3.2.2.8 b15

```
DWORD StructBitfield32::b15
```

Bit 15.

3.2.2.9 b16

```
DWORD StructBitfield32::b16
```

Bit 16.

3.2.2.10 b17

```
DWORD StructBitfield32::b17
```

Bit 17.

3.2.2.11 b18

```
DWORD StructBitfield32::b18
```

Bit 18.

3.2.2.12 b19

```
DWORD StructBitfield32::b19
```

Bit 19.

3.2.2.13 b2

```
DWORD StructBitfield32::b2
```

Bit 2.

3.2.2.14 b20

`DWORD StructBitfield32::b20`

Bit 20.

3.2.2.15 b21

`DWORD StructBitfield32::b21`

Bit 21.

3.2.2.16 b22

`DWORD StructBitfield32::b22`

Bit 22.

3.2.2.17 b23

`DWORD StructBitfield32::b23`

Bit 23.

3.2.2.18 b24

`DWORD StructBitfield32::b24`

Bit 24.

3.2.2.19 b25

`DWORD StructBitfield32::b25`

Bit 25.

3.2.2.20 b26

`DWORD StructBitfield32::b26`

Bit 26.

3.2.2.21 b27

```
DWORD StructBitfield32::b27
```

Bit 27.

3.2.2.22 b28

```
DWORD StructBitfield32::b28
```

Bit 28.

3.2.2.23 b29

```
DWORD StructBitfield32::b29
```

Bit 29.

3.2.2.24 b3

```
DWORD StructBitfield32::b3
```

Bit 3.

3.2.2.25 b30

```
DWORD StructBitfield32::b30
```

Bit 30.

3.2.2.26 b31

```
DWORD StructBitfield32::b31
```

Bit 31 (MSB)

3.2.2.27 b4

```
DWORD StructBitfield32::b4
```

Bit 4.

3.2.2.28 b5

```
DWORD StructBitfield32::b5
```

Bit 5.

3.2.2.29 b6

```
DWORD StructBitfield32::b6
```

Bit 6.

3.2.2.30 b7

```
DWORD StructBitfield32::b7
```

Bit 7.

3.2.2.31 b8

```
DWORD StructBitfield32::b8
```

Bit 8.

3.2.2.32 b9

```
DWORD StructBitfield32::b9
```

Bit 9.

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

- arm_typedefs.h

3.3 StructBitfield64 Struct Reference

Bitfield 64b.

```
#include <arm_typedefs.h>
```

Public Attributes

- LWORD b0:1

  Bit 0 (LSB)
- LWORD b1:1

  Bit 1.
- LWORD b2:1

  Bit 2.
- LWORD b3:1

  Bit 3.
- LWORD b4:1

  Bit 4.
- LWORD b5:1

  Bit 5.
- LWORD b6:1

  Bit 6.
- LWORD b7:1

  Bit 7.
- LWORD b8:1

  Bit 8.
- LWORD b9:1

  Bit 9.
- LWORD b10:1

  Bit 10.
- LWORD b11:1

  Bit 11.
- LWORD b12:1

  Bit 12.
- LWORD b13:1

  Bit 13.
- LWORD b14:1

  Bit 14.
- LWORD b15:1

  Bit 15.
- LWORD b16:1

  Bit 16.
- LWORD b17:1

  Bit 17.
- LWORD b18:1

  Bit 18.
- LWORD b19:1

  Bit 19.
- LWORD b20:1

  Bit 20.
- LWORD b21:1

  Bit 21.
- LWORD b22:1

  Bit 22.
- LWORD b23:1

  Bit 23.
- LWORD b24:1

- LWORD b50:1

  Bit 50.
- LWORD b51:1

  Bit 51.
- LWORD b52:1

  Bit 52.
- LWORD b53:1

  Bit 53.
- LWORD b54:1

  Bit 54.
- LWORD b55:1

  Bit 55.
- LWORD b56:1

  Bit 56.
- LWORD b57:1

  Bit 57.
- LWORD b58:1

  Bit 58.
- LWORD b59:1

  Bit 59.
- LWORD b60:1

  Bit 60.
- LWORD b61:1

  Bit 61.
- LWORD b62:1

  Bit 62.
- LWORD b63:1

  Bit 63 (MSB)

3.3.1 Detailed Description

Bitfield 64b.

3.3.2 Member Data Documentation

3.3.2.1 b0

```
LWORD StructBitfield64::b0
```

Bit 0 (LSB)

3.3.2.2 b1

```
LWORD StructBitfield64::b1
```

Bit 1.

3.3.2.3 b10

```
LWORD StructBitfield64::b10
```

Bit 10.

3.3.2.4 b11

```
LWORD StructBitfield64::b11
```

Bit 11.

3.3.2.5 b12

```
LWORD StructBitfield64::b12
```

Bit 12.

3.3.2.6 b13

```
LWORD StructBitfield64::b13
```

Bit 13.

3.3.2.7 b14

```
LWORD StructBitfield64::b14
```

Bit 14.

3.3.2.8 b15

```
LWORD StructBitfield64::b15
```

Bit 15.

3.3.2.9 b16

```
LWORD StructBitfield64::b16
```

Bit 16.

3.3.2.10 b17

`LWORD StructBitfield64::b17`

Bit 17.

3.3.2.11 b18

`LWORD StructBitfield64::b18`

Bit 18.

3.3.2.12 b19

`LWORD StructBitfield64::b19`

Bit 19.

3.3.2.13 b2

`LWORD StructBitfield64::b2`

Bit 2.

3.3.2.14 b20

`LWORD StructBitfield64::b20`

Bit 20.

3.3.2.15 b21

`LWORD StructBitfield64::b21`

Bit 21.

3.3.2.16 b22

`LWORD StructBitfield64::b22`

Bit 22.

3.3.2.17 b23

`LWORD StructBitfield64::b23`

Bit 23.

3.3.2.18 b24

`LWORD StructBitfield64::b24`

Bit 24.

3.3.2.19 b25

`LWORD StructBitfield64::b25`

Bit 25.

3.3.2.20 b26

`LWORD StructBitfield64::b26`

Bit 26.

3.3.2.21 b27

`LWORD StructBitfield64::b27`

Bit 27.

3.3.2.22 b28

`LWORD StructBitfield64::b28`

Bit 28.

3.3.2.23 b29

`LWORD StructBitfield64::b29`

Bit 29.

3.3.2.24 b3

```
LWORD StructBitfield64::b3
```

Bit 3.

3.3.2.25 b30

```
LWORD StructBitfield64::b30
```

Bit 30.

3.3.2.26 b31

```
LWORD StructBitfield64::b31
```

Bit 31.

3.3.2.27 b32

```
LWORD StructBitfield64::b32
```

Bit 32.

3.3.2.28 b33

```
LWORD StructBitfield64::b33
```

Bit 33.

3.3.2.29 b34

```
LWORD StructBitfield64::b34
```

Bit 34.

3.3.2.30 b35

```
LWORD StructBitfield64::b35
```

Bit 35.

3.3.2.31 b36

```
LWORD StructBitfield64::b36
```

Bit 36.

3.3.2.32 b37

```
LWORD StructBitfield64::b37
```

Bit 37.

3.3.2.33 b38

```
LWORD StructBitfield64::b38
```

Bit 38.

3.3.2.34 b39

```
LWORD StructBitfield64::b39
```

Bit 39.

3.3.2.35 b4

```
LWORD StructBitfield64::b4
```

Bit 4.

3.3.2.36 b40

```
LWORD StructBitfield64::b40
```

Bit 40.

3.3.2.37 b41

```
LWORD StructBitfield64::b41
```

Bit 41.

3.3.2.38 b42

```
LWORD StructBitfield64::b42
```

Bit 42.

3.3.2.39 b43

```
LWORD StructBitfield64::b43
```

Bit 43.

3.3.2.40 b44

```
LWORD StructBitfield64::b44
```

Bit 44.

3.3.2.41 b45

```
LWORD StructBitfield64::b45
```

Bit 45.

3.3.2.42 b46

```
LWORD StructBitfield64::b46
```

Bit 46.

3.3.2.43 b47

```
LWORD StructBitfield64::b47
```

Bit 47.

3.3.2.44 b48

```
LWORD StructBitfield64::b48
```

Bit 48.

3.3.2.45 b49

`LWORD StructBitfield64::b49`

Bit 49.

3.3.2.46 b5

`LWORD StructBitfield64::b5`

Bit 5.

3.3.2.47 b50

`LWORD StructBitfield64::b50`

Bit 50.

3.3.2.48 b51

`LWORD StructBitfield64::b51`

Bit 51.

3.3.2.49 b52

`LWORD StructBitfield64::b52`

Bit 52.

3.3.2.50 b53

`LWORD StructBitfield64::b53`

Bit 53.

3.3.2.51 b54

`LWORD StructBitfield64::b54`

Bit 54.

3.3.2.52 b55

`LWORD StructBitfield64::b55`

Bit 55.

3.3.2.53 b56

`LWORD StructBitfield64::b56`

Bit 56.

3.3.2.54 b57

`LWORD StructBitfield64::b57`

Bit 57.

3.3.2.55 b58

`LWORD StructBitfield64::b58`

Bit 58.

3.3.2.56 b59

`LWORD StructBitfield64::b59`

Bit 59.

3.3.2.57 b6

`LWORD StructBitfield64::b6`

Bit 6.

3.3.2.58 b60

`LWORD StructBitfield64::b60`

Bit 60.

3.3.2.59 b61

`LWORD StructBitfield64::b61`

Bit 61.

3.3.2.60 b62

`LWORD StructBitfield64::b62`

Bit 62.

3.3.2.61 b63

`LWORD StructBitfield64::b63`

Bit 63 (MSB)

3.3.2.62 b7

`LWORD StructBitfield64::b7`

Bit 7.

3.3.2.63 b8

`LWORD StructBitfield64::b8`

Bit 8.

3.3.2.64 b9

`LWORD StructBitfield64::b9`

Bit 9.

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

- arm_typedefs.h

3.4 StructBitfield8 Struct Reference

Bitfield 8b.

```
#include <arm_typedefs.h>
```

Public Attributes

- BYTE b0:1

  Bit 0 (LSB)
- BYTE b1:1

  Bit 1.
- BYTE b2:1

  Bit 2.
- BYTE b3:1

  Bit 3.
- BYTE b4:1

  Bit 4.
- BYTE b5:1

  Bit 5.
- BYTE b6:1

  Bit 6.
- BYTE b7:1

  Bit 7 (MSB)

3.4.1 Detailed Description

Bitfield 8b.

3.4.2 Member Data Documentation

3.4.2.1 b0

```
BYTE StructBitfield8::b0
```

Bit 0 (LSB)

3.4.2.2 b1

```
BYTE StructBitfield8::b1
```

Bit 1.

3.4.2.3 b2

```
BYTE StructBitfield8::b2
```

Bit 2.

3.4.2.4 b3

```
BYTE StructBitfield8::b3
```

Bit 3.

3.4.2.5 b4

```
BYTE StructBitfield8::b4
```

Bit 4.

3.4.2.6 b5

```
BYTE StructBitfield8::b5
```

Bit 5.

3.4.2.7 b6

```
BYTE StructBitfield8::b6
```

Bit 6.

3.4.2.8 b7

```
BYTE StructBitfield8::b7
```

Bit 7 (MSB)

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

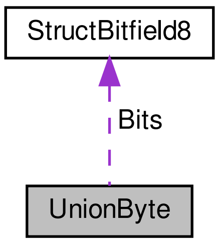
- arm_typedefs.h

3.5 UnionByte Union Reference

Union for BYTE.

```
#include <arm_typedefs.h>
```

Collaboration diagram for UnionByte:



Public Attributes

- BYTE Byte

  BYTE.
- sBitfield8 Bits

  Bits.

3.5.1 Detailed Description

Union for BYTE.

3.5.2 Member Data Documentation

3.5.2.1 Bits

```
sBitfield8 UnionByte::Bits
```

Bits.

3.5.2.2 Byte

```
BYTE UnionByte::Byte
```

BYTE.

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

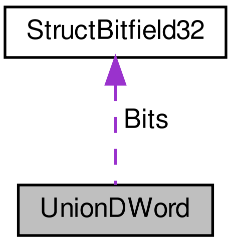
- arm_typedefs.h

3.6 UnionDWord Union Reference

Union for DWORD.

```
#include <arm_typedefs.h>
```

Collaboration diagram for UnionDWord:



Public Attributes

- DWORD DWord

  32b
- WORD Word [2]

  Words tab.
- BYTE Byte [4]

  Bytes tab.
- struct {
  WORD W0:16
  W0 LSWord.
  WORD W1:16
  W1 MSWord.
  } Words

- struct {
  BYTE B0:8
  B0 LSByte.
  BYTE B1:8
  B1.
  BYTE B2:8
  B2.
  BYTE B3:8
  B3 MSByte.
  } Bytes
- sBitfield32 Bits
  Bits.

3.6.1 Detailed Description

Union for DWORD.

3.6.2 Member Data Documentation

3.6.2.1 B0

```
BYTE UnionDWord::B0
```

B0 LSByte.

3.6.2.2 B1

```
BYTE UnionDWord::B1
```

B1.

3.6.2.3 B2

```
BYTE UnionDWord::B2
```

B2.

3.6.2.4 B3

```
BYTE UnionDWord::B3
```

B3 MSByte.

3.6.2.5 Bits

```
sBitfield32 UnionDWord::Bits
```

Bits.

3.6.2.6 Byte

```
BYTE UnionDWord::Byte[4]
```

Bytes tab.

3.6.2.7 Bytes

```
struct { ... } UnionDWord::Bytes
```

3.6.2.8 DWord

```
DWORD UnionDWord::DWord
```

32b

3.6.2.9 W0

```
WORD UnionDWord::W0
```

W0 LSWord.

3.6.2.10 W1

```
WORD UnionDWord::W1
```

W1 MSWord.

3.6.2.11 Word

```
WORD UnionDWord::Word[2]
```

Words tab.

3.6.2.12 Words

```
struct { ...  } UnionDWord::Words
```

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

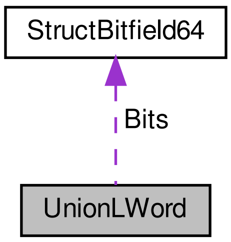
- arm_typedefs.h

3.7 UnionLWord Union Reference

Union for LWORD.

```
#include <arm_typedefs.h>
```

Collaboration diagram for UnionLWord:



Public Attributes

- LWORD LWord

  64b
- DWORD DWord [2]

  DWords tab.
- WORD Word [4]

  Words tab.
- BYTE Byte [8]

  Bytes tab.
- struct {
  DWORD D0:32
  DW0 LSDWord.
  DWORD D1:32
  DW1 MSDWord.
  } DWords

- struct {
  WORD W0:16
  W0 LSWord.
  WORD W1:16
  W1.
  WORD W2:16
  W2.
  WORD W3:16
  W3 MSWord.
  } Words

- struct {
  BYTE B0:8
  B0 LSByte.
  BYTE B1:8
  B1.
  BYTE B2:8
  B2.
  BYTE B3:8
  B3.
  BYTE B4:8
  B4.
  BYTE B5:8
  B5.
  BYTE B6:8
  B6.
  BYTE B7:8
  B7 MSByte.
  } Bytes

- sBitfield64 Bits

  Bits.

3.7.1 Detailed Description

Union for LWORD.

3.7.2 Member Data Documentation

3.7.2.1 B0

```
BYTE UnionLWord::B0
```

B0 LSByte.

3.7.2.2 B1

```
BYTE UnionLWord::B1
```

B1.

3.7.2.3 B2

```
BYTE UnionLWord::B2
```

B2.

3.7.2.4 B3

```
BYTE UnionLWord::B3
```

B3.

3.7.2.5 B4

```
BYTE UnionLWord::B4
```

B4.

3.7.2.6 B5

```
BYTE UnionLWord::B5
```

B5.

3.7.2.7 B6

```
BYTE UnionLWord::B6
```

B6.

3.7.2.8 B7

```
BYTE UnionLWord::B7
```

B7 MSByte.

3.7.2.9 Bits

```
sBitfield64 UnionLWord::Bits
```

Bits.

3.7.2.10 Byte

```
BYTE UnionLWord::Byte[8]
```

Bytes tab.

3.7.2.11 Bytes

```
struct { ... } UnionLWord::Bytes
```

3.7.2.12 D0

```
DWORD UnionLWord::D0
```

DW0 LSDWord.

3.7.2.13 D1

```
DWORD UnionLWord::D1
```

DW1 MSDWord.

3.7.2.14 DWord

```
DWORD UnionLWord::DWord[2]
```

DWords tab.

3.7.2.15 DWords

```
struct { ... } UnionLWord::DWords
```

3.7.2.16 LWord

```
LWORD UnionLWord::LWord
```

64b

3.7.2.17 W0

```
WORD UnionLWord::W0
```

W0 LSWord.

3.7.2.18 W1

```
WORD UnionLWord::W1
```

W1.

3.7.2.19 W2

```
WORD UnionLWord::W2
```

W2.

3.7.2.20 W3

```
WORD UnionLWord::W3
```

W3 MSWord.

3.7.2.21 Word

```
WORD UnionLWord::Word[4]
```

Words tab.

3.7.2.22 Words

```
struct { ...  } UnionLWord::Words
```

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

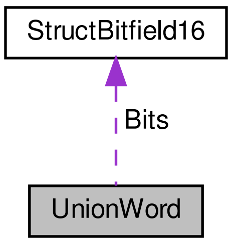
- arm_typedefs.h

3.8 UnionWord Union Reference

Union for WORD.

```
#include <arm_typedefs.h>
```

Collaboration diagram for UnionWord:



Public Attributes

- WORD Word

  16b
- BYTE Byte [2]

  Bytes tab.
- struct {
    BYTE B0:8
      LSByte.
    BYTE B1:8
      MSByte.
  } Bytes
- sBitfield16 Bits

  Bits.

3.8.1 Detailed Description

Union for WORD.

3.8.2 Member Data Documentation

3.8.2.1 B0

```
BYTE UnionWord::B0
```

LSByte.

3.8.2.2 B1

```
BYTE UnionWord::B1
```

MSByte.

3.8.2.3 Bits

```
sBitfield16 UnionWord::Bits
```

Bits.

3.8.2.4 Byte

```
BYTE UnionWord::Byte[2]
```

Bytes tab.

3.8.2.5 Bytes

```
struct { ... } UnionWord::Bytes
```

3.8.2.6 Word

```
WORD UnionWord::Word
```

16b

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

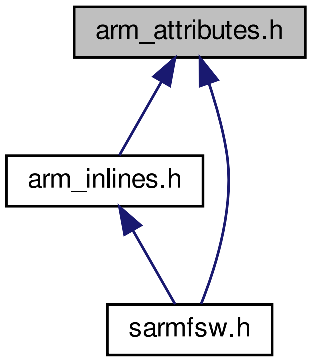
- arm_typedefs.h

4 File Documentation

4.1 arm_attributes.h File Reference

ARM common gcc attributes.

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



Macros

- #define INLINE__ __attribute__((always_inline))

  Inline *attribute for gcc*
- #define WEAK__ __attribute__((weak))

  Weak *attribute for gcc*
- #define PACK__ __attribute__((__packed__))

  Packed *attribute for gcc*

4.1.1 Detailed Description

ARM common gcc attributes.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.1.2 Macro Definition Documentation

4.1.2.1 INLINE__

```
#define INLINE__ __attribute__((always_inline))
```

Inline attribute for gcc

4.1.2.2 PACK__

```
#define PACK__ __attribute__((__packed__))
```

Packed attribute for gcc

4.1.2.3 WEAK__

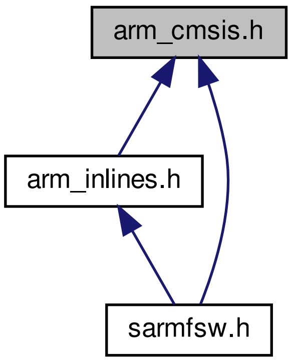
```
#define WEAK__ __attribute__((weak))
```

Weak attribute for gcc

4.2 arm_cmsis.h File Reference

ARM link with CMSIS files.

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



Macros

- #define diInterrupts() __disable_irq()

  Disable interruptions macro.
- #define enInterrupts() __enable_irq()

  Enable interruptions macro.

4.2.1 Detailed Description

ARM link with CMSIS files.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.2.2 Macro Definition Documentation

4.2.2.1 diInterrupts

```
#define diInterrupts( ) __disable_irq()
```

Disable interruptions macro.

4.2.2.2 enInterrupts

```
#define enInterrupts( ) __enable_irq()
```

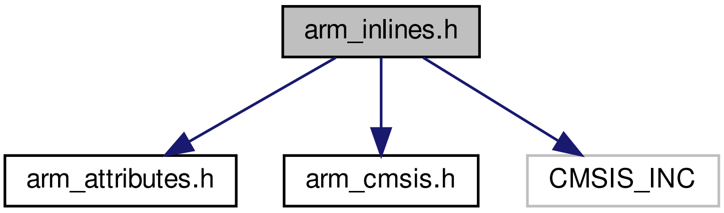
Enable interruptions macro.

4.3 arm_inlines.h File Reference

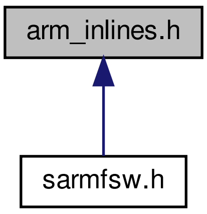
ARM common inlines.

```
#include "arm_attributes.h"
#include "arm_cmsis.h"
#include <CMSIS_INC>
```

Include dependency graph for arm_inlines.h:



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



Functions

- bool TPSSUP_MS (uint32_t val, uint32_t time)

  Tests if stored time value has reached time lapse in ms.
- bool TPSINF_MS (uint32_t val, uint32_t time)

  Tests if stored time value has not reached time lapse in ms.
- uint16_t conv8upto16Bits (uint8_t val, uint8_t nb)

  converts 8bits to 8+nb bits (16bits max)
- uint32_t conv16upto32Bits (uint16_t val, uint8_t nb)

  converts 16bits to 16+nb bits (32bits max)
- uint16_t SWAP_END16B (uint16_t w)

  Swap endians of the contents of a 16b value.
- uint32_t SWAP_END32B (uint32_t d)

Swap endians of the contents of a 32b value.
- void SWAP_END16B_TAB (uint16_t tab[ ], uint16_t nb)

  Swap endians of a 16b tab.
- void SWAP_END32B_TAB (uint32_t tab[ ], uint16_t nb)

  Swap endians of a 32b tab.
- bool inTolerance (int32_t val, int32_t ref, int32_t tolerance)

  Checks if val given as parameter is in tolerance.
- bool inRange (int32_t val, int32_t low, int32_t high)

  Checks if val given as parameter is in range.

4.3.1 Detailed Description

ARM common inlines.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.3.2 Function Documentation

4.3.2.1 conv16upto32Bits()

```
uint32_t conv16upto32Bits (
            uint16_t val,
            uint8_t nb )  [inline]
```

converts 16bits to 16+nb bits (32bits max)

Warning

conversion output shall not exceed 32bits (input shall strictly be unsigned 16bits)
nb shall be in range 0-16 (note that using 0 doesn't change val)

Parameters

in	*val*	- 16b value to convert
in	*nb*	- number of bits to add (16bits max)

Returns

Converted value

4.3.2.2 conv8upto16Bits()

```
uint16_t conv8upto16Bits (
            uint8_t val,
            uint8_t nb )  [inline]
```

converts 8bits to 8+nb bits (16bits max)

Warning

conversion output shall not exceed 16bits (input shall strictly be unsigned 8bits)
nb shall be in range 0-8 (note that using 0 doesn't change val)

Parameters

in	*val*	- 8b value to convert
in	*nb*	- number of bits to add (8bits max)

Returns

Converted value

4.3.2.3 inRange()

```
bool inRange (
            int32_t val,
            int32_t low,
            int32_t high )  [inline]
```

Checks if val given as parameter is in range.

Parameters

in	*val*	- Value to check
in	*low*	- Low range boundary
in	*high*	- High range boundary

Returns

true if val is inRange

4.3.2.4 inTolerance()

```
bool inTolerance (
             int32_t val,
             int32_t ref,
             int32_t tolerance )  [inline]
```

Checks if val given as parameter is in tolerance.

Parameters

in	*val*	- Value to check
in	*ref*	- Reference value
in	*tolerance*	- Tolerance on reference value (in percent)

Returns

true if val is inTolerance

4.3.2.5 SWAP_END16B()

```
uint16_t SWAP_END16B (
             uint16_t w )  [inline]
```

Swap endians of the contents of a 16b value.

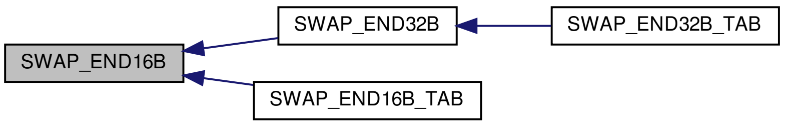
Parameters

in	*w*	- 16b value

Returns

Swapped value

Here is the caller graph for this function:



4.3.2.6 SWAP_END16B_TAB()

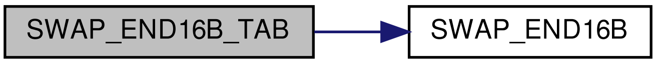
```
void SWAP_END16B_TAB (
             uint16_t tab[],
             uint16_t nb )  [inline]
```

Swap endians of a 16b tab.

Parameters

in	*tab*	- tab of 16b values
in	*nb*	- nb of values in tab

Here is the call graph for this function:



4.3.2.7 SWAP_END32B()

```
uint32_t SWAP_END32B (
             uint32_t d )  [inline]
```

Swap endians of the contents of a 32b value.

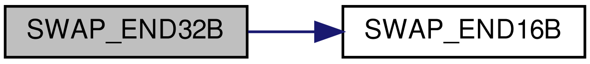
Parameters

in	*d*	- 32b value

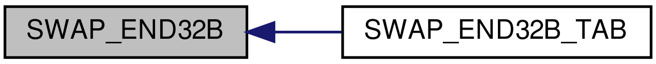
Returns

Swapped value

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.8 SWAP_END32B_TAB()

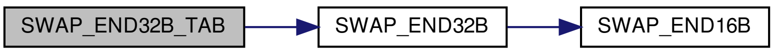
```
void SWAP_END32B_TAB (
            uint32_t tab[],
            uint16_t nb )  [inline]
```

Swap endians of a 32b tab.

Parameters

in	*tab*	- tab of 32b values
in	*nb*	- nb of values in tab

Here is the call graph for this function:



4.3.2.9 TPSINF_MS()

```
bool TPSINF_MS (
            uint32_t val,
            uint32_t time )  [inline]
```

Tests if stored time value has not reached time lapse in ms.

Parameters

in	*val*	- stored time value
in	*time*	- time lapse (in ms)

Returns

true if time not elapsed

4.3.2.10 TPSSUP_MS()

```
bool TPSSUP_MS (
            uint32_t val,
            uint32_t time )  [inline]
```

Tests if stored time value has reached time lapse in ms.

Parameters

in	*val*	- stored time value
in	*time*	- time lapse (in ms)

Returns

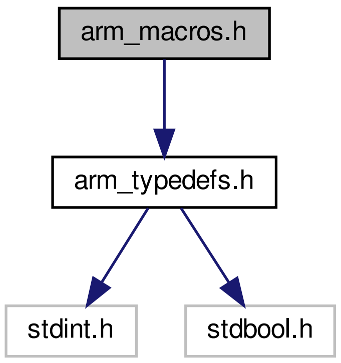
true if time elapsed

4.4 arm_macros.h File Reference

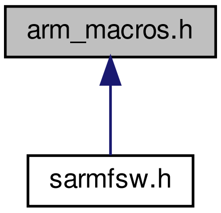
ARM common macros.

`#include "arm_typedefs.h"`

Include dependency graph for arm_macros.h:



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



Macros

- #define Undefined -1

  Undefined value.
- #define Null 0

  Null Value.
- #define pNull (void ∗) 0

  Null pointer -> same as NULL in Stdlib.h.
- #define charNUL '\0'

  Null Char.
- #define MAKEWORD(b1, b2) ((WORD) (((BYTE) (b1)) | ((WORD) ((BYTE) (b2))) ∗ 0x100))

  *Make WORD from **b1** and **b2** with **b1** as LSB.*
- #define MAKELONG(w1, w2) ((DWORD) (((WORD) (w1)) | ((DWORD) ((WORD) (w2))) ∗ 0x10000))

  *Make LONG from **w1** and **w2** with **w1** as LSB.*
- #define LOWORD(l) ((WORD) (l))

  *Get WORD LSW from LONG **l**.*
- #define HIWORD(l) ((WORD) ((DWORD) (l) / 0x10000))

  *Get WORD MSW from LONG **l**.*
- #define LOBYTE(w) ((BYTE) (w))

  *Get BYTE LSB from WORD **w**.*
- #define HIBYTE(w) ((BYTE) ((WORD) (w) / 0x100))

  *Get BYTE MSB from WORD **w**.*
- #define BYTE_TO_PERC(b) ((BYTE) (((b) ∗ 100) / 255))

  *Converts a BYTE **b** (0-255) to percent (0-100)*
- #define PERC_TO_BYTE(p) ((BYTE) (((p) > 100 ? 100 : (p)) ∗ 255 / 100))

  *Converts a BYTE **p** percentage (0-100) to BYTE (0-255) with max checking.*
- #define OFFSET_OF(typ, mbr) ((size_t) &(((typ∗)0)->mbr))

  *Computes the offset member **mbr** from struct **typ**.*
- #define SZ_OBJ(obj, typ) (sizeof(obj) / sizeof(typ))

  *Computes the number of elements of **obj** following **typ**.*
- #define CAT(a, b) a##b

  Preprocessor Name concatenation.
- #define XCAT(a, b) CAT(a, b)

  Preprocessor Name concatenation (possible nesting)
- #define STR(s) ("" #s)

Stringify an expression.

- #define binEval(exp) ((exp) ? true : false)

  *boolean evaluation of expression **exp***
- #define nbinEval(exp) (!binEval(exp))

  *complemented boolean evaluation of expression **exp***
- #define max(a, b) ((a) >= (b) ? (a) : (b))

  *Returns max value between **a** and **b**.*
- #define min(a, b) ((a) <= (b) ? (a) : (b))

  *Returns min value between **a** and **b**.*
- #define MIN3(a, b, c) ((b) <= (c) ? ((a) <= (b) ? (a) : (b)) : ((a) <= (c) ? (a) : (c)))

  *Returns max value between **a**, **b** and **c**.*
- #define MAX3(a, b, c) ((b) >= (c) ? ((a) >= (b) ? (a) : (b)) : ((a) >= (c) ? (a) : (c)))

  *Returns min value between **a**, **b** and **c**.*
- #define CLAMP(v, min, max) ((v) < (min) ? (min) : ((v) > (max) ? (max) : (v)))

  *Returns the value between **min** and **max** from **val**.*
- #define OneThird ((float) (1.0 / 3.0))

  1/3 approximation
- #define TwoThird ((float) (2.0 / 3.0))

  2/3 approximation
- #define Pi 3.141593f

  Approximate Pi calculation (4 ∗ atan(1))
- #define RADIAN_TO_FLOAT(r) ((float) (((r) > 2∗Pi ? 2∗Pi : (r)) / 2∗Pi))
- #define FLOAT_TO_RADIAN(f) ((float) ((((f) > 1.0f ? 1.0f : (f)) < 0.0f ? 0.0f : (f)) ∗ 2∗Pi)
- #define DEGREE_TO_FLOAT(d) ((float) (((d) > 360.0f ? 360.0f : (d)) / 360.0f))
- #define FLOAT_TO_DEGREE(f) ((float) ((((f) > 1.0f ? 1.0f : (f)) < 0.0f ? 0.0f : (f)) ∗ 360.0f))
- #define SWAP_BYTE(a, b) { BYTE c; c = a; a = b; b = c; }

  *Swap BYTEs **a** & **b**.*
- #define SWAP_WORD(a, b) { WORD c; c = a; a = b; b = c; }

  *Swap WORDs **a** & **b**.*
- #define SWAP_DWORD(a, b) { DWORD c; c = a; a = b; b = c; }

  *Swap DWORDs **a** & **b**.*

4.4.1 Detailed Description

ARM common macros.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.4.2 Macro Definition Documentation

4.4.2.1 binEval

```
#define binEval(
              exp ) ((exp) ? true : false)
```

boolean evaluation of expression **exp**

4.4.2.2 BYTE_TO_PERC

```
#define BYTE_TO_PERC(
              b ) ((BYTE) (((b) * 100) / 255))
```

Converts a BYTE **b** (0-255) to percent (0-100)

4.4.2.3 CAT

```
#define CAT(
              a,
              b ) a##b
```

Preprocessor Name concatenation.

Warning

No nesting possible, use *XCAT* in this case

4.4.2.4 charNUL

```
#define charNUL '\0'
```

Null Char.

4.4.2.5 CLAMP

```
#define CLAMP(
              v,
              min,
              max ) ((v) < (min) ? (min) : ((v) > (max) ? (max) : (v)))
```

Returns the value between **min** and **max** from **val**.

4.4.2.6 DEGREE_TO_FLOAT

```
#define DEGREE_TO_FLOAT(
              d ) ((float) (((d) > 360.0f ? 360.0f : (d)) / 360.0f))
```

4.4.2.7 FLOAT_TO_DEGREE

```
#define FLOAT_TO_DEGREE(
              f ) ((float) ((((f) > 1.0f ? 1.0f : (f)) < 0.0f ? 0.0f : (f)) * 360.0f))
```

4.4.2.8 FLOAT_TO_RADIAN

```
#define FLOAT_TO_RADIAN(
              f ) ((float) ((((f) > 1.0f ? 1.0f : (f)) < 0.0f ? 0.0f : (f)) * 2*Pi)
```

4.4.2.9 HIBYTE

```
#define HIBYTE(
              w ) ((BYTE) ((WORD) (w) / 0x100))
```

Get BYTE MSB from WORD **w**.

4.4.2.10 HIWORD

```
#define HIWORD(
              l ) ((WORD) ((DWORD) (l) / 0x10000))
```

Get WORD MSW from LONG **l**.

4.4.2.11 LOBYTE

```
#define LOBYTE(
              w ) ((BYTE) (w))
```

Get BYTE LSB from WORD **w**.

4.4.2.12 LOWORD

```
#define LOWORD(
              l ) ((WORD) (l))
```

Get WORD LSW from LONG **l**.

4.4.2.13 MAKELONG

```
#define MAKELONG(
              w1,
              w2 ) ((DWORD) (((WORD) (w1)) | ((DWORD) ((WORD) (w2))) * 0x10000))
```

Make LONG from **w1** and **w2** with **w1** as LSB.

4.4.2.14 MAKEWORD

```
#define MAKEWORD(
              b1,
              b2 ) ((WORD) (((BYTE) (b1)) | ((WORD) ((BYTE) (b2))) * 0x100))
```

Make WORD from **b1** and **b2** with **b1** as LSB.

4.4.2.15 max

```
#define max(
              a,
              b ) ((a) >= (b) ? (a) : (b))
```

Returns max value between **a** and **b**.

4.4.2.16 MAX3

```
#define MAX3(
              a,
              b,
              c ) ((b) >= (c) ? ((a) >= (b) ? (a) : (b)) : ((a) >= (c) ? (a) : (c)))
```

Returns min value between **a**, **b** and **c**.

4.4.2.17 min

```
#define min(
              a,
              b ) ((a) <= (b) ? (a) : (b))
```

Returns min value between **a** and **b**.

4.4.2.18 MIN3

```
#define MIN3(
              a,
              b,
              c ) ((b) <= (c) ? ((a) <= (b) ? (a) : (b)) : ((a) <= (c) ? (a) : (c)))
```

Returns max value between **a**, **b** and **c**.

4.4.2.19 nbinEval

```
#define nbinEval(
              exp ) (!binEval(exp))
```

complemented boolean evaluation of expression **exp**

4.4.2.20 Null

```
#define Null 0
```

Null Value.

4.4.2.21 OFFSET_OF

```
#define OFFSET_OF(
              typ,
              mbr ) ((size_t) &(((typ*)0)->mbr))
```

Computes the offset member **mbr** from struct **typ**.

4.4.2.22 OneThird

```
#define OneThird ((float) (1.0 / 3.0))
```

1/3 approximation

4.4.2.23 PERC_TO_BYTE

```
#define PERC_TO_BYTE(
              p ) ((BYTE) (((p) > 100 ? 100 : (p)) * 255 / 100))
```

Converts a BYTE **p** percentage (0-100) to BYTE (0-255) with max checking.

4.4.2.24 Pi

```
#define Pi 3.141593f
```

Approximate Pi calculation (4 ∗ atan(1))

4.4.2.25 pNull

```
#define pNull (void *) 0
```

Null pointer -> same as NULL in Stdlib.h.

4.4.2.26 RADIAN_TO_FLOAT

```
#define RADIAN_TO_FLOAT(
              r ) ((float) (((r) > 2*Pi ? 2*Pi : (r)) / 2*Pi))
```

4.4.2.27 STR

```
#define STR(
              s ) ("" #s)
```

Stringify an expression.

4.4.2.28 SWAP_BYTE

```
#define SWAP_BYTE(
              a,
              b ) { BYTE c; c = a; a = b; b = c; }
```

Swap BYTEs **a** & **b**.

4.4.2.29 SWAP_DWORD

```
#define SWAP_DWORD(
              a,
              b ) { DWORD c; c = a; a = b; b = c; }
```

Swap DWORDs **a** & **b**.

4.4.2.30 SWAP_WORD

```
#define SWAP_WORD(
              a,
              b ) { WORD c; c = a; a = b; b = c; }
```

Swap WORDs **a** & **b**.

4.4.2.31 SZ_OBJ

```
#define SZ_OBJ(
              obj,
              typ ) (sizeof(obj) / sizeof(typ))
```

Computes the number of elements of **obj** following **typ**.

4.4.2.32 TwoThird

```
#define TwoThird ((float) (2.0 / 3.0))
```

2/3 approximation

4.4.2.33 Undefined

```
#define Undefined -1
```

Undefined value.

4.4.2.34 XCAT

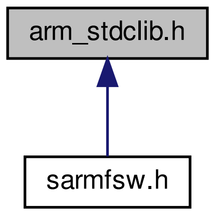
```
#define XCAT(
              a,
              b ) CAT(a, b)
```

Preprocessor Name concatenation (possible nesting)

4.5 arm_stdclib.h File Reference

ARM common standard c library wrapper macros.

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



Macros

- #define printExpr(e) (printf("%s = %d\r\n", #e, (e)))

  *Print expression **e** and it's result **e** using printf.*
- #define verbInstr(i) (printf("" #i), (i))

  *Print instruction **e** and execute it.*
- #define str_clr(s) (s[0] = '\0')

  *clear string **s** (fast way)*
- #define str_clr_safe(s) (memset('\0', s, sizeof(s)))

  *clear string **s** (safe way)*
- #define str_add_tab(s) (strcat(s, '\t'))

  Adding tab to string using strcat.
- #define str_add_cr(s) (strcat(s, '\r\n'))

  Adding new line to string using strcat.
- #define VerboseInc(x) (puts("Incrementing " #x), (x)++)

  Increment example using puts.
- #define TestMalloc(x) ((x) = malloc(sizeof(∗x)), assert(x))

  Asserted malloc.

4.5.1 Detailed Description

ARM common standard c library wrapper macros.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.5.2 Macro Definition Documentation

4.5.2.1 printExpr

```
#define printExpr(
              e ) (printf("%s = %d\r\n", #e, (e)))
```

Print expression **e** and it's result **e** using printf.

4.5.2.2 str_add_cr

```
#define str_add_cr(
              s ) (strcat(s, '\r\n'))
```

Adding new line to string using strcat.

4.5.2.3 str_add_tab

```
#define str_add_tab(
              s ) (strcat(s, '\t'))
```

Adding tab to string using strcat.

4.5.2.4 str_clr

```
#define str_clr(
              s ) (s[0] = '\0')
```

clear string **s** (fast way)

4.5.2.5 str_clr_safe

```
#define str_clr_safe(
              s ) (memset('\0', s, sizeof(s)))
```

clear string **s** (safe way)

4.5.2.6 TestMalloc

```
#define TestMalloc(
              x ) ((x) = malloc(sizeof(*x)), assert(x))
```

Asserted malloc.

4.5.2.7 verbInstr

```
#define verbInstr(
              i ) (printf("" #i), (i))
```

Print instruction **e** and execute it.

4.5.2.8 VerboseInc

```
#define VerboseInc(
              x ) (puts("Incrementing " #x), (x)++)
```

Increment example using puts.

4.6 arm_stm32.h File Reference

ARM common macros for STM32.

Macros

- #define port(mnem) XCAT(mnem, _GPIO_Port)

  Wrapper for PORT Alias.
- #define pin(mnem) XCAT(mnem, _Pin)

  Wrapper for PIN Alias.
- #define gpio(mnem) port(mnem), pin(mnem)

  Wrapper for PORT/PIN Alias (when using HAL_GPIO_ReadPin for example)
- #define STM_HEADER(f) XCAT(<stm32, XCAT(f, xx.h>))

  *concatenate <stm32(f)xx.h> name following stm family **f***
- #define STM_CONF_HEADER(f) XCAT(<stm32, XCAT(f, xx_hal.h>))

  *concatenate <stm32(f)xx_hal.h> name following stm family **f***
- #define STM32_INC STM_HEADER(STM_FAMILY)

  Alias for STM32 include.
- #define STM32_CFG STM_CONF_HEADER(STM_FAMILY)

  Alias for STM32 include.

4.6.1 Detailed Description

ARM common macros for STM32.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.6.2 Macro Definition Documentation

4.6.2.1 gpio

```
#define gpio(
              mnem ) port(mnem), pin(mnem)
```

Wrapper for PORT/PIN Alias (when using HAL_GPIO_ReadPin for example)

4.6.2.2 pin

```
#define pin(
              mnem ) XCAT(mnem, _Pin)
```

Wrapper for PIN Alias.

4.6.2.3 port

```
#define port(
              mnem ) XCAT(mnem, _GPIO_Port)
```

Wrapper for PORT Alias.

4.6.2.4 STM32_CFG

```
#define STM32_CFG STM_CONF_HEADER(STM_FAMILY)
```

Alias for STM32 include.

4.6.2.5 STM32_INC

```
#define STM32_INC STM_HEADER(STM_FAMILY)
```

Alias for STM32 include.

4.6.2.6 STM_CONF_HEADER

```
#define STM_CONF_HEADER(
              f ) XCAT(<stm32, XCAT(f, xx_hal.h>))
```

concatenate <stm32(f)xx_hal.h> name following stm family **f**

4.6.2.7 STM_HEADER

```
#define STM_HEADER(
              f ) XCAT(<stm32, XCAT(f, xx.h>))
```

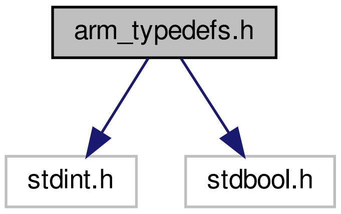
concatenate <stm32(f)xx.h> name following stm family **f**

4.7 arm_typedefs.h File Reference

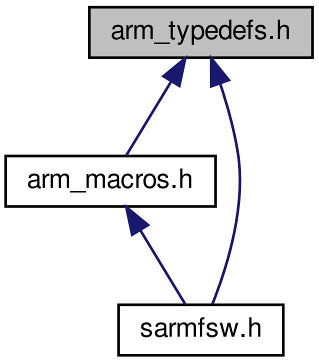
ARM common typedefs.

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

Include dependency graph for arm_typedefs.h:



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



Classes

- struct StructBitfield8

  Bitfield 8b.
- struct StructBitfield16

  Bitfield 16b.
- struct StructBitfield32

  Bitfield 32b.
- struct StructBitfield64

  Bitfield 64b.
- union UnionByte

  Union for BYTE.
- union UnionWord

  Union for WORD.
- union UnionDWord

  Union for DWORD.
- union UnionLWord

  Union for LWORD.

Typedefs

- typedef char CHAR

  Char typedef (8bits)
- typedef uint8_t BYTE

  Unsigned Byte typedef (8bits)
- typedef uint16_t WORD

  Unsigned Word typedef (16bits)
- typedef uint32_t DWORD

  Unsigned dWord typedef (32bits)
- typedef uint64_t LWORD

  Unsigned lWord typedef (64bits)
- typedef int8_t SBYTE

  Signed Byte typedef (8bits)
- typedef int16_t SWORD

  Signed Word typedef (16bits)
- typedef int32_t SDWORD

  Signed dWord typedef (32bits)
- typedef int64_t SLWORD

  Signed lWord typedef (64bits)
- typedef enum eState eState
- typedef enum eEdge eEdge
- typedef struct StructBitfield8 sBitfield8
- typedef struct StructBitfield16 sBitfield16
- typedef struct StructBitfield32 sBitfield32
- typedef struct StructBitfield64 sBitfield64
- typedef union UnionByte uByte
- typedef union UnionWord uWord
- typedef union UnionDWord uDWord
- typedef union UnionLWord uLWord

Enumerations

- enum eState { Off = 0U, On = 1U }

    Activation state On, Off.
- enum eEdge { NoEdge = 0, Rising, Falling }

    Signal Edges.

4.7.1 Detailed Description

ARM common typedefs.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.7.2 Typedef Documentation

4.7.2.1 BYTE

```
typedef uint8_t BYTE
```

Unsigned Byte typedef (8bits)

4.7.2.2 CHAR

```
typedef char CHAR
```

Char typedef (8bits)

4.7.2.3 DWORD

```
typedef uint32_t DWORD
```

Unsigned dWord typedef (32bits)

4.7.2.4 eEdge

```
typedef enum eEdge eEdge
```

4.7.2.5 eState

```
typedef enum eState eState
```

4.7.2.6 LWORD

```
typedef uint64_t LWORD
```

Unsigned lWord typedef (64bits)

4.7.2.7 sBitfield16

```
typedef struct StructBitfield16 sBitfield16
```

4.7.2.8 sBitfield32

```
typedef struct StructBitfield32 sBitfield32
```

4.7.2.9 sBitfield64

```
typedef struct StructBitfield64 sBitfield64
```

4.7.2.10 sBitfield8

```
typedef struct StructBitfield8 sBitfield8
```

4.7.2.11 SBYTE

```
typedef int8_t SBYTE
```

Signed Byte typedef (8bits)

4.7.2.12 SDWORD

```
typedef int32_t SDWORD
```

Signed dWord typedef (32bits)

4.7.2.13 SLWORD

```
typedef int64_t SLWORD
```

Signed lWord typedef (64bits)

4.7.2.14 SWORD

```
typedef int16_t SWORD
```

Signed Word typedef (16bits)

4.7.2.15 uByte

```
typedef union UnionByte uByte
```

4.7.2.16 uDWord

```
typedef union UnionDWord uDWord
```

4.7.2.17 uLWord

```
typedef union UnionLWord uLWord
```

4.7.2.18 uWord

```
typedef union UnionWord uWord
```

4.7.2.19 WORD

```
typedef uint16_t WORD
```

Unsigned Word typedef (16bits)

4.7.3 Enumeration Type Documentation

4.7.3.1 eEdge

```
enum eEdge
```

Signal Edges.

Enumerator

NoEdge	No change.
Rising	Rising edge.
Falling	Falling edge.

4.7.3.2 eState

```
enum eState
```

Activation state On, Off.

Enumerator

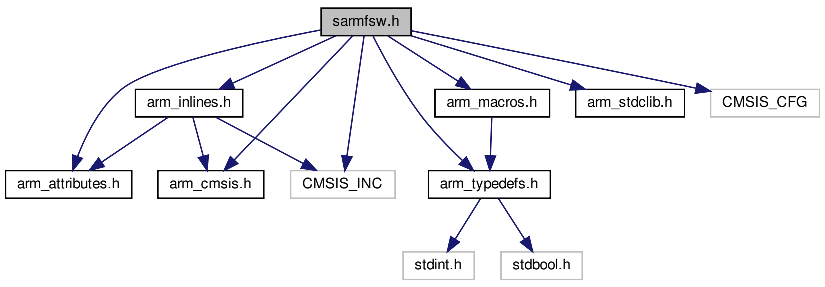
Off	Off / Clear.
On	On / Set.

4.8 sarmfsw.h File Reference

ARM common headers for projects.

```
#include "arm_attributes.h"
#include "arm_typedefs.h"
#include "arm_macros.h"
#include "arm_stdclib.h"
#include "arm_cmsis.h"
#include <CMSIS_INC>
#include <CMSIS_CFG>
#include "arm_inlines.h"
```

Include dependency graph for sarmfsw.h:



Typedefs

- typedef enum FW_target FW_target

Enumerations

- enum FW_target {
  DefSpecialTarget = 0, DefDebugTarget, DefReleaseTarget, DefFUBARTarget,
  DefUnknownTarget = 0xFF }

  Firmware target types.

4.8.1 Detailed Description

ARM common headers for projects.

Author

SMFSW

Date

2017

Copyright

MIT (c) 2017, SMFSW

4.8.2 Typedef Documentation

4.8.2.1 FW_target

```
typedef enum FW_target FW_target
```

4.8.3 Enumeration Type Documentation

4.8.3.1 FW_target

```
enum FW_target
```

Firmware target types.

Enumerator

DefSpecialTarget	Special FW target (same as debug, yet)
DefDebugTarget	Debug FW target (default)
DefReleaseTarget	Release FW target (No debug information)
DefFUBARTarget	FUBAR FW target (shall be used only for stress/testing purposes)
DefUnknownTarget	Unknown FW target (should never happen!)

Index