sarmfsw: SMFSW Toolbox (for ARM & compatible with Arduino platform)

3.2

Generated by Doxygen 1.8.13

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

3 Data Structure Documentation

3.1 StructBitfield16 Struct Reference

Bitfield 16b.

```
#include "arm_typedefs.h"
```

Data Fields

- 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 Field 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"
```

Data Fields

- 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 Field 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"
```

Data Fields

- 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.

Bit 41.
- LWORD b42:1

  Bit 42.
- LWORD b43:1

  Bit 43.
- LWORD b44:1

  Bit 44.
- LWORD b45:1

  Bit 45.
- LWORD b46:1

  Bit 46.
- LWORD b47:1

  Bit 47.
- LWORD b48:1

  Bit 48.
- LWORD b49:1

  Bit 49.
- 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 Field 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"
```

Data Fields

- 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 Field 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:

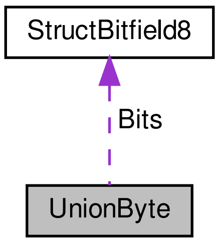
- arm_typedefs.h

3.5 UnionByte Union Reference

Union for BYTE.

```
#include "arm_typedefs.h"
```

Collaboration diagram for UnionByte:



Data Fields

- BYTE Byte

  BYTE.
- sBitfield8 Bits

  Bits.

3.5.1 Detailed Description

Union for BYTE.

3.5.2 Field 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:

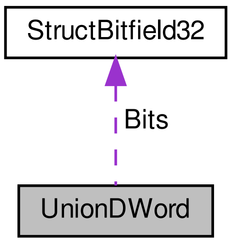
- arm_typedefs.h

3.6 UnionDWord Union Reference

Union for DWORD.

`#include "arm_typedefs.h"`

Collaboration diagram for UnionDWord:



Data Fields

- DWORD DWord

  32b
- WORD Word [2]

  Words tab.
- BYTE Byte [4]

  Bytes tab.
- struct {
  WORD W1:16
  W1 MSWord.
  WORD W0:16
  W0 LSWord.
  } Words
- struct {
  BYTE B3:8
  B3 MSByte.
  BYTE B2:8
  B2.
  BYTE B1:8
  B1.
  BYTE B0:8
  B0 LSByte.
  } Bytes
- sBitfield32 Bits

  Bits.

3.6.1 Detailed Description

Union for DWORD.

3.6.2 Field 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:

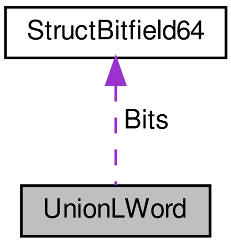
- arm_typedefs.h

3.7 UnionLWord Union Reference

Union for LWORD.

```
#include "arm_typedefs.h"
```

Collaboration diagram for UnionLWord:



Data Fields

- LWORD LWord

  64b
- DWORD DWord [2]

  DWords tab.
- WORD Word [4]

  Words tab.
- BYTE Byte [8]

  Bytes tab.
- struct {
  DWORD D1:32
  DW1 MSDWord.
  DWORD D0:32
  DW0 LSDWord.
  } DWords
- struct {
  WORD W3:16
  W3 MSWord.
  WORD W2:16
  W2.
  WORD W1:16
  W1.
  WORD W0:16
  W0 LSWord.
  } Words
- struct {
  BYTE B7:8
  B7 MSByte.
  BYTE B6:8
  B6.
  BYTE B5:8
  B5.
  BYTE B4:8
  B4.
  BYTE B3:8
  B3.
  BYTE B2:8
  B2.
  BYTE B1:8
  B1.
  BYTE B0:8
  B0 LSByte.
  } Bytes
- sBitfield64 Bits

  Bits.

3.7.1 Detailed Description

Union for LWORD.

3.7.2 Field 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:

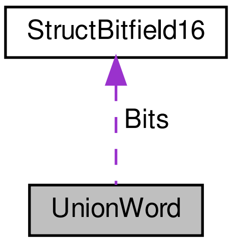
- arm_typedefs.h

3.8 UnionWord Union Reference

Union for WORD.

```
#include "arm_typedefs.h"
```

Collaboration diagram for UnionWord:



Data Fields

- WORD Word

  16b
- BYTE Byte [2]

  Bytes tab.
- struct {

  BYTE B1:8

  MSByte.

  BYTE B0:8

  LSByte.

  } Bytes
- sBitfield16 Bits

  Bits.

3.8.1 Detailed Description

Union for WORD.

3.8.2 Field 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:

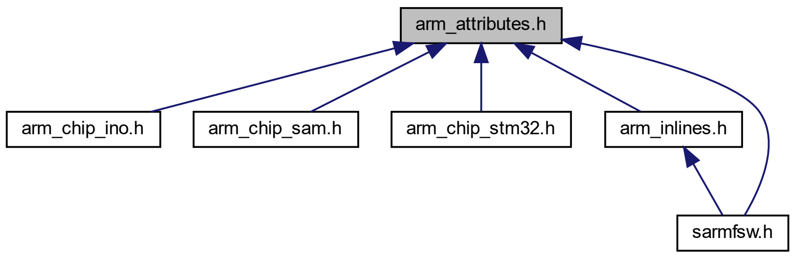
- arm_typedefs.h

4 File Documentation

4.1 arm_attributes.h File Reference

ARM common compilers attributes.

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



Macros

- #define __WEAK __attribute__((weak))

    ***Weak** attribute*
- #define __IRQ __attribute__((interrupt_handler))

    ***Interrupt** attribute*
- #define ALIGN__(n) __attribute__((align(n)))

    Align** attribute padded to **n
- #define COLD__ __attribute__((cold))

    ***Cold** attribute*
- #define DEPRECATED__ __attribute__((deprecated))

    ***Deprecated** attribute*
- #define HOT__ __attribute__((hot))

    ***Hot** attribute*
- #define INLINE__ __attribute__((always_inline))

    ***Always inline** attribute*
- #define NONNULL__ __attribute__((nonnull))

    ***Non null** attribute (all pointers will be checked)*
- #define NORETURN__ __attribute__((noreturn))

    ***No return** attribute*
- #define PACK__ __attribute__((__packed__))

    ***Packed** attribute*
- #define PURE__ __attribute__((pure))

    ***Pure** attribute*

4.1.1 Detailed Description

ARM common compilers attributes.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.1.2 Macro Definition Documentation

4.1.2.1 __IRQ

```
#define __IRQ __attribute__((interrupt_handler))
```

Interrupt attribute

4.1.2.2 __WEAK

```
#define __WEAK __attribute__((weak))
```

Weak attribute

4.1.2.3 ALIGN__

```
#define ALIGN__(
              n ) __attribute__((align(n)))
```

Align attribute padded to **n**

4.1.2.4 COLD__

```
#define COLD__ __attribute__((cold))
```

Cold attribute

4.1.2.5 DEPRECATED__

```
#define DEPRECATED__ __attribute__((deprecated))
```

Deprecated attribute

4.1.2.6 HOT__

```
#define HOT__ __attribute__((hot))
```

Hot attribute

4.1.2.7 INLINE__

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

Always inline attribute

4.1.2.8 NONNULL__

```
#define NONNULL__ __attribute__((nonnull))
```

Non null attribute (all pointers will be checked)

4.1.2.9 NORETURN__

```
#define NORETURN__ __attribute__((noreturn))
```

No return attribute

4.1.2.10 PACK__

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

Packed attribute

4.1.2.11 PURE__

```
#define PURE__ __attribute__((pure))
```

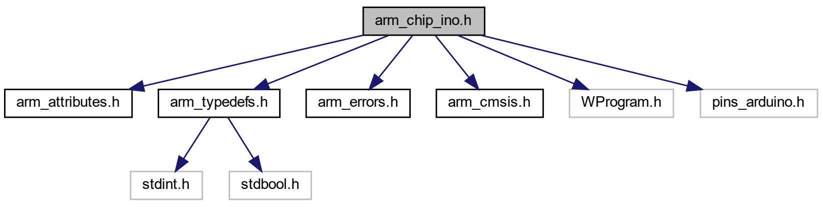
Pure attribute

4.2 arm_chip_ino.h File Reference

Common macros for Arduino.

```
#include "arm_attributes.h"
#include "arm_typedefs.h"
#include "arm_errors.h"
#include "arm_cmsis.h"
#include "WProgram.h"
#include "pins_arduino.h"
```

Include dependency graph for arm_chip_ino.h:



Macros

- #define diInterrupts() noInterrupts()

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

    Enable interruptions macro.
- #define HAL_MAX_TICKS ((uint32_t) -1)

    Max Ticks value.
- #define HAL_MS_TICKS_FACTOR 1

    Milliseconds multiplier (depending tick counter frequency)
- #define HALTicks() millis()

    Alias for Arduino get ms ticks function.

Functions

- FctERR HALERRtoFCTERR (int32_t status)

    Convert Arduino error code to FctERR.

4.2.1 Detailed Description

Common macros for Arduino.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.2.2 Macro Definition Documentation

4.2.2.1 diInterrupts

```
#define diInterrupts( ) noInterrupts()
```

Disable interruptions macro.

4.2.2.2 enInterrupts

```
#define enInterrupts( ) interrupts()
```

Enable interruptions macro.

4.2.2.3 HAL_MAX_TICKS

```
#define HAL_MAX_TICKS ((uint32_t) -1)
```

Max Ticks value.

Note

Define HAL_MAX_TICKS with custom max value in project if tick max value is not using 32b variable full scale

4.2.2.4 HAL_MS_TICKS_FACTOR

```
#define HAL_MS_TICKS_FACTOR 1
```

Milliseconds multiplier (depending tick counter frequency)

Note

Define HAL_MS_TICKS_FACTOR with custom multiplier in project if tick period is not 1ms

4.2.2.5 HALTicks

```
#define HALTicks( ) millis()
```

Alias for Arduino get ms ticks function.

4.2.3 Function Documentation

4.2.3.1 HALERRtoFCTERR()

```
FctERR HALERRtoFCTERR (
            int32_t status )  [inline]
```

Convert Arduino error code to FctERR.

Parameters

in	*status*	- Arduino error code

Returns

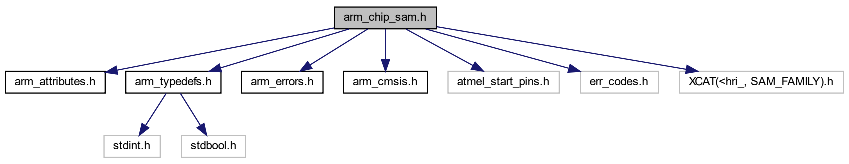
FctERR status

4.3 arm_chip_sam.h File Reference

ARM common macros for Atmel SAM families.

```
#include "arm_attributes.h"
#include "arm_typedefs.h"
#include "arm_errors.h"
#include "arm_cmsis.h"
#include "atmel_start_pins.h"
#include "err_codes.h"
#include <XCAT(<hri_, SAM_FAMILY).h>
```

Include dependency graph for arm_chip_sam.h:



Macros

- #define SAM_HEADER(f) XCAT(<hri_, f).h>

  *concatenate <hri_(f).h> name following sam family **f***
- #define SAM_CONF_HEADER(f) <sam.h>

  *<sam.h> name following sam family **f***
- #define ARM_CMSIS_INC SAM_HEADER(SAM_FAMILY)

  Alias for SAM CMSIS include.
- #define ARM_HAL_CFG SAM_CONF_HEADER(SAM_FAMILY)

  Alias for SAM HAL config include.
- #define HAL_MAX_TICKS ((uint32_t) -1)

  HAL max Ticks value.
- #define HAL_MS_TICKS_FACTOR 1

  HAL milliseconds multiplier (depending tick counter frequency)
- #define HALTicks() HAL_GetTick()

  Alias for HAL get ticks function.

Functions

- FctERR HALERRtoFCTERR (int32_t status)

  Convert ATMEL error code to FctERR.

4.3.1 Detailed Description

ARM common macros for Atmel SAM families.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

Attention

On SAM families you should configure a timer to count for ms. A TIM peripheral shall be configured in ATMEL START (with a period of 1ms). Using driver examples from ATMEL START generated code, you can add this code to your projects.

```
static struct timer_task TIMER_0_task1;
static volatile uint32_t uwTick = 0;


uint32_t HAL_GetTick(void) {        // Declare HALTicks() at project level if you're using a different
       getter function name
    return uwTick; }

static void TIMER_0_task1_cb(const struct timer_task *const timer_task) {
    uwTick++; }

void TIMER_0_start(void)            // Adapt function if TIM configured is not TIMER_0
{
    TIMER_0_task1.interval = 1;     // Adjust interval if TIM period is faster than 1ms (or define
       appropriate HAL_MS_TICKS_FACTOR)
    TIMER_0_task1.cb = TIMER_0_task1_cb;
    TIMER_0_task1.mode = TIMER_TASK_REPEAT;

    timer_add_task(&TIMER_0, &TIMER_0_task1);
    timer_start(&TIMER_0);
}
```

Please note TIMER_0_start() shall be called at init. Also, HAL_GetTick shall be known to sarmfsw. As atmel_↩ start_pins.h is included by sarmfsw, you should add HAL_GetTick prototype in the file:

```
#include <stdint.h>
uint32_t HAL_GetTick(void);
```

4.3.2 Macro Definition Documentation

4.3.2.1 ARM_CMSIS_INC

```
#define ARM_CMSIS_INC SAM_HEADER(SAM_FAMILY)
```

Alias for SAM CMSIS include.

4.3.2.2 ARM_HAL_CFG

```
#define ARM_HAL_CFG SAM_CONF_HEADER(SAM_FAMILY)
```

Alias for SAM HAL config include.

4.3.2.3 HAL_MAX_TICKS

```
#define HAL_MAX_TICKS ((uint32_t) -1)
```

HAL max Ticks value.

Note

Define HAL_MAX_TICKS with custom max value in project if tick max value is not using 32b variable full scale

4.3.2.4 HAL_MS_TICKS_FACTOR

```
#define HAL_MS_TICKS_FACTOR 1
```

HAL milliseconds multiplier (depending tick counter frequency)

Note

Define HAL_MS_TICKS_FACTOR with custom multiplier in project if tick period is not 1ms

4.3.2.5 HALTicks

```
#define HALTicks( ) HAL_GetTick()
```

Alias for HAL get ticks function.

Note

Define HALTicks at project level to call your own ms tick getter function

4.3.2.6 SAM_CONF_HEADER

```
#define SAM_CONF_HEADER(
              f ) <sam.h>
```

<sam.h> name following sam family **f**

4.3.2.7 SAM_HEADER

```
#define SAM_HEADER(
              f ) XCAT(<hri_, f).h>
```

concatenate <hri_(f).h> name following sam family **f**

4.3.3 Function Documentation

4.3.3.1 HALERRtoFCTERR()

```
FctERR HALERRtoFCTERR (
             int32_t status ) [inline]
```

Convert ATMEL error code to FctERR.

Parameters

in	*status*	- ATMEL error code

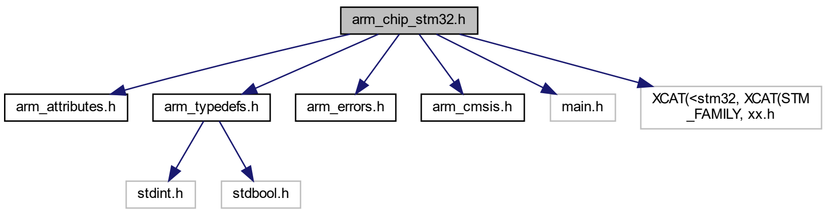
Returns

FctERR status

4.4 arm_chip_stm32.h File Reference

ARM common macros for STM32.

```
#include "arm_attributes.h"
#include "arm_typedefs.h"
#include "arm_errors.h"
#include "arm_cmsis.h"
#include "main.h"
#include <XCAT(<stm32, XCAT(STM_FAMILY, xx.h>
```
Include dependency graph for arm_chip_stm32.h:



Macros

- #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 ARM_CMSIS_INC STM_HEADER(STM_FAMILY)

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

    Alias for STM32 HAL config include.
- #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 timer(mnem) XCAT(mnem, _Tim)

    Wrapper for TIM Alias.
- #define channel(mnem) XCAT(mnem, _Chan)
- #define TIM(mnem) timer(mnem), channel(mnem)

Wrapper for TIM/CHAN Alias (when using HAL_TIM_PWM_Start for example)
- #define HAL_MAX_TICKS ((uint32_t) -1)

  HAL max Ticks value.
- #define HAL_MS_TICKS_FACTOR 1

  HAL milliseconds multiplier (depending tick counter frequency)
- #define HALTicks() HAL_GetTick()

  Alias for HAL get ticks function.

Functions

- FctERR HALERRtoFCTERR (HAL_StatusTypeDef status)

  Convert HAL_StatusTypeDef to FctERR.

4.4.1 Detailed Description

ARM common macros for STM32.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.4.2 Macro Definition Documentation

4.4.2.1 ARM_CMSIS_INC

```
#define ARM_CMSIS_INC STM_HEADER(STM_FAMILY)
```

Alias for STM32 CMSIS include.

4.4.2.2 ARM_HAL_CFG

```
#define ARM_HAL_CFG STM_CONF_HEADER(STM_FAMILY)
```

Alias for STM32 HAL config include.

4.4.2.3 channel

```
#define channel(
              mnem ) XCAT(mnem, _Chan)
```

Wrapper for TIM Channel Alias

4.4.2.4 GPIO

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

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

4.4.2.5 HAL_MAX_TICKS

```
#define HAL_MAX_TICKS ((uint32_t) -1)
```

HAL max Ticks value.

Note

Define HAL_MAX_TICKS with custom max value in project if tick max value is not using 32b variable full scale

4.4.2.6 HAL_MS_TICKS_FACTOR

```
#define HAL_MS_TICKS_FACTOR 1
```

HAL milliseconds multiplier (depending tick counter frequency)

Note

Define HAL_MS_TICKS_FACTOR with custom multiplier in project if tick period is not 1ms

4.4.2.7 HALTicks

```
#define HALTicks( ) HAL_GetTick()
```

Alias for HAL get ticks function.

4.4.2.8 pin

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

Wrapper for PIN Alias.

4.4.2.9 port

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

Wrapper for PORT Alias.

4.4.2.10 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**

4.4.2.11 STM_HEADER

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

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

4.4.2.12 TIM

```
#define TIM(
              mnem ) timer(mnem), channel(mnem)
```

Wrapper for TIM/CHAN Alias (when using HAL_TIM_PWM_Start for example)

Note

You would have to define mnemonic _Tim/_Chan corresponding to what's defined in CubeMX as Port/Pin (for consistency)

4.4.2.13 timer

```
#define timer(
              mnem ) XCAT(mnem, _Tim)
```

Wrapper for TIM Alias.

4.4.3 Function Documentation

4.4.3.1 HALERRtoFCTERR()

```
FctERR HALERRtoFCTERR (
             HAL_StatusTypeDef status )  [inline]
```

Convert HAL_StatusTypeDef to FctERR.

Parameters

in	*status*	- HAL_StatusTypeDef status

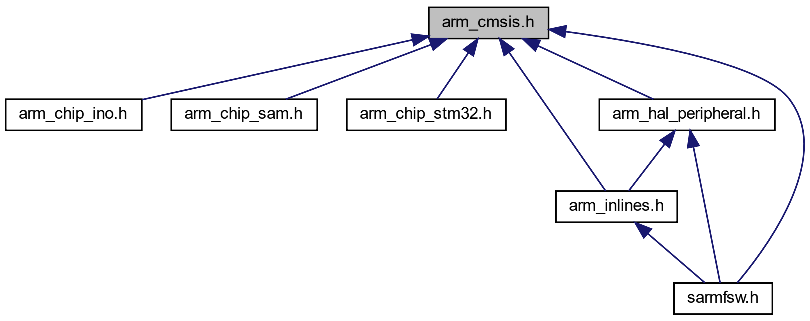
Returns

FctERR status

4.5 arm_cmsis.h File Reference

ARM link with CMSIS files.

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



Macros

- #define dilnterrupts() __disable_irq()

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

  Enable interruptions macro.

4.5.1 Detailed Description

ARM link with CMSIS files.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.5.2 Macro Definition Documentation

4.5.2.1 diInterrupts

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

Disable interruptions macro.

4.5.2.2 enInterrupts

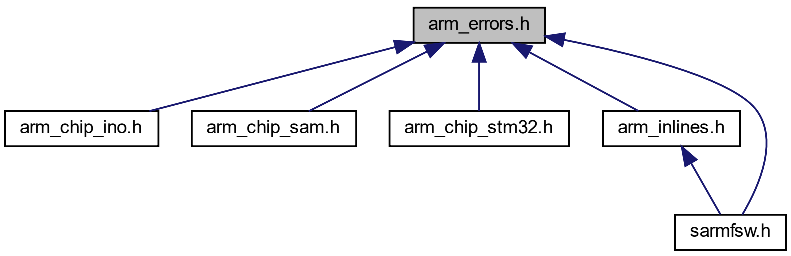
```
#define enInterrupts( ) __enable_irq()
```

Enable interruptions macro.

4.6 arm_errors.h File Reference

ARM user errors.

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



Typedefs

- typedef enum FctERR FctERR

Enumerations

- enum FctERR {
ERROR_OK = 0, ERROR_SPEED = -1, ERROR_RANGE = -2, ERROR_TIMEOUT = -3,
ERROR_VALUE = -4, ERROR_OVERFLOW = -5, ERROR_MATH = -6, ERROR_ENABLED = -7,
ERROR_DISABLED = -8, ERROR_BUSY = -9, ERROR_NOTAVAIL = -10, ERROR_RXEMPTY = -11,
ERROR_TXFULL = -12, ERROR_BUSOFF = -13, ERROR_OVERRUN = -14, ERROR_FRAMING = -15,
ERROR_PARITY = -16, ERROR_NOISE = -17, ERROR_IDLE = -18, ERROR_FAULT = -19,
ERROR_BREAK = -20, ERROR_CRC = -21, ERROR_ARBITR = -22, ERROR_PROTECT = -23,
ERROR_UNDERFLOW = -24, ERROR_UNDERRUN = -25, ERROR_COMMON = -26, ERROR_LINSYNC = -27,
ERROR_FAILED = -28, ERROR_QFULL = -29, ERROR_CMD = -30, ERROR_NOTIMPLEM = -31,
ERROR_MEMORY = -32, ERROR_INSTANCE = -33 }

  Enum of low/mid level functions return state.

4.6.1 Detailed Description

ARM user errors.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.6.2 Typedef Documentation

4.6.2.1 FctERR

```
typedef enum FctERR FctERR
```

4.6.3 Enumeration Type Documentation

4.6.3.1 FctERR

```
enum FctERR
```

Enum of low/mid level functions return state.

Enumerator

ERROR_OK	OK.
ERROR_SPEED	This device does not work in the active speed mode.
ERROR_RANGE	Parameter out of range.
ERROR_TIMEOUT	Abort on timeout error.
ERROR_VALUE	Parameter of incorrect value.
ERROR_OVERFLOW	Overflow.
ERROR_MATH	Overflow during evaluation.
ERROR_ENABLED	Device is enabled.
ERROR_DISABLED	Device is disabled.
ERROR_BUSY	Device is busy.
ERROR_NOTAVAIL	Requested value or method not available.
ERROR_RXEMPTY	No data in receiver.
ERROR_TXFULL	Transmitter is full.
ERROR_BUSOFF	Bus not available.
ERROR_OVERRUN	Overrun error is detected.
ERROR_FRAMING	Framing error is detected.
ERROR_PARITY	Parity error is detected.
ERROR_NOISE	Noise error is detected.

Enumerator

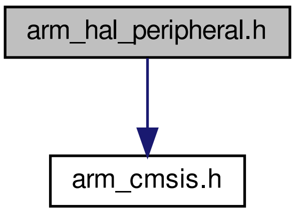
ERROR_IDLE	Idle error is detected.
ERROR_FAULT	Fault error is detected.
ERROR_BREAK	Break char is received during communication.
ERROR_CRC	CRC error is detected.
ERROR_ARBITR	A node lost arbitration. This error occurs if two nodes start transmission at the same time.
ERROR_PROTECT	Protection error is detected.
ERROR_UNDERFLOW	Underflow error is detected.
ERROR_UNDERRUN	Underrun error is detected.
ERROR_COMMON	Common error of a device.
ERROR_LINSYNC	LIN synchronization error is detected.
ERROR_FAILED	Requested functionality or process failed.
ERROR_QFULL	Queue is full.
ERROR_CMD	Command error is detected.
ERROR_NOTIMPLEM	Function not implemented error.
ERROR_MEMORY	Memory error.
ERROR_INSTANCE	Instance error.

4.7 arm_hal_peripheral.h File Reference

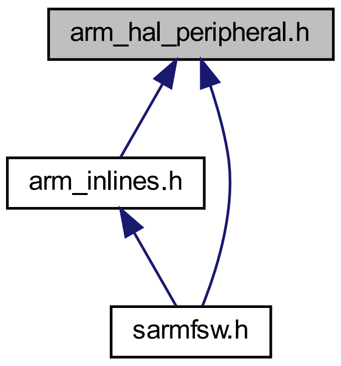
ARM HAL peripheral includes.

```
#include "arm_cmsis.h"
```

Include dependency graph for arm_hal_peripheral.h:



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



4.7.1 Detailed Description

ARM HAL peripheral includes.

Warning

for STM32, HAL shall be configured to generate as pairs of h/c files

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

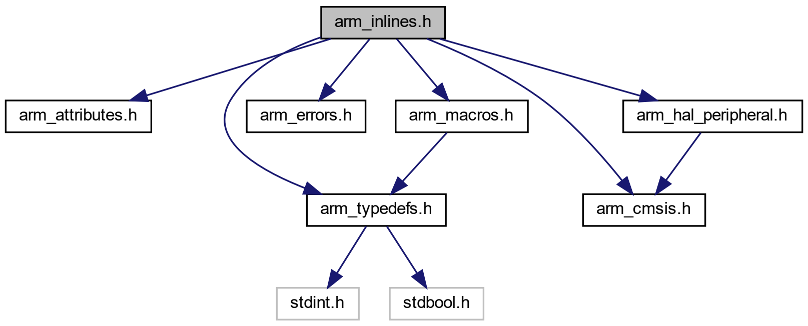
4.8 arm_inlines.h File Reference

ARM common inlines.

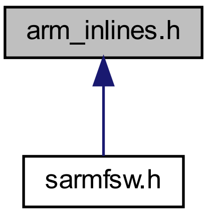
```
#include "arm_attributes.h"
#include "arm_typedefs.h"
#include "arm_errors.h"
#include "arm_macros.h"
#include "arm_cmsis.h"
```

```
#include "arm_hal_peripheral.h"
```

Include dependency graph for arm_inlines.h:



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



Functions

- bool INLINE__ TPSSUP_MS (const DWORD last, const DWORD time)

  Tests if stored time value has reached time lapse in ms.
- bool INLINE__ TPSINF_MS (const DWORD last, const DWORD time)

  Tests if stored time value has not reached time lapse in ms.
- BYTE HexToBCD (const BYTE hex)

  Converts hexadecimal value to BCD.
- BYTE BCDToHex (const BYTE bcd)

  Converts BCD value to hexadecimal.
- CHAR INLINE__ HexToASCII (BYTE hex)

  Converts hexadecimal value to ASCII.
- SBYTE ASCIIToHex (const CHAR ascii)

  Converts ASCII char to hexadecimal.
- DWORD INLINE__ bin2gray (const DWORD bin)

  Convert binary value to gray code.
- DWORD gray2bin (const DWORD gray)

  Convert gray code to binary value.

- BYTE INLINE__ conv16to8Bits (const WORD val)

  converts 16bits to 8bits
- WORD INLINE__ conv8to16Bits (const BYTE val)

  converts 8bits to 16bits
- WORD conv8upto16Bits (const BYTE val, const BYTE nb)

  converts 8bits to 8+nb bits (16bits max)
- DWORD conv16upto32Bits (const WORD val, const BYTE nb)

  converts 16bits to 16+nb bits (32bits max)
- LWORD conv32upto64Bits (const DWORD val, const BYTE nb)

  converts 32bits to 32+nb bits (64bits max)
- eEndian testEndian_basic (void)

  Test Core endian.
- eEndian testEndian_full (void)

  Test Core endian (full, recognizing mid endians too)
- WORD SWAP_END16B (const WORD w)

  Swap endians of the contents of a 16b value (little -> big, big -> little)
- DWORD SWAP_END32B (const DWORD d)

  Swap endians of the contents of a 32b value (little -> big, big -> little)
- LWORD SWAP_END64B (const LWORD l)

  Swap endians of the contents of a 64b value (little -> big, big -> little)
- void INLINE__ SWAP_END16B_TAB (WORD tab[ ], const WORD nb)

  Swap endians of a 16b tab (little -> big, big -> little)
- void INLINE__ SWAP_END32B_TAB (DWORD tab[ ], const WORD nb)

  Swap endians of a 32b tab (little -> big, big -> little)
- void INLINE__ SWAP_END64B_TAB (LWORD tab[ ], const WORD nb)

  Swap endians of a 64b tab (little -> big, big -> little)
- bool inTolerance (const SDWORD val, const SDWORD ref, float tolerance)

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

  Checks if val given as parameter is in range.
- int32_t get_fp_dec (float f, uint8_t nb)

  Get floating point number decimal part.

4.8.1 Detailed Description

ARM common inlines.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.8.2 Function Documentation

4.8.2.1 ASCIIToHex()

```
SBYTE ASCIIToHex (
                const CHAR ascii )  [inline]
```

Converts ASCII char to hexadecimal.

Parameters

in	*ascii*	- ASCII char to convert

Returns

Hexadecimal value

4.8.2.2 BCDToHex()

```
BYTE BCDToHex (
            const BYTE bcd )  [inline]
```

Converts BCD value to hexadecimal.

Note

Returns 0xFF if BCD value is inconsistent

Parameters

in	*bcd*	- BCD value to convert

Returns

Hexadecimal value

4.8.2.3 bin2gray()

```
DWORD INLINE__ bin2gray (
            const DWORD bin )  [inline]
```

Convert binary value to gray code.

Parameters

in	*bin*	- binary value

Returns

Converted value (gray code)

4.8.2.4 conv16to8Bits()

```
BYTE INLINE__ conv16to8Bits (
            const WORD val )  [inline]
```

converts 16bits to 8bits

Parameters

in	*val*	- 16b value to convert

Returns

Converted value

4.8.2.5 conv16upto32Bits()

```
DWORD conv16upto32Bits (
            const WORD val,
            const BYTE 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

4.8.2.6 conv32upto64Bits()

```
LWORD conv32upto64Bits (
            const DWORD val,
            const BYTE nb )  [inline]
```

converts 32bits to 32+nb bits (64bits max)

Warning

conversion output shall not exceed 64bits (input shall strictly be unsigned 32bits)
nb shall be in range 0-32 (note that using 0 doesn't change val)

Parameters

in	*val*	- 32b value to convert
in	*nb*	- number of bits to add (32bits max)

Returns

Converted value

4.8.2.7 conv8to16Bits()

```
WORD INLINE__ conv8to16Bits (
            const BYTE val )  [inline]
```

converts 8bits to 16bits

Parameters

in	*val*	- 8b value to convert

Returns

Converted value

4.8.2.8 conv8upto16Bits()

```
WORD conv8upto16Bits (
            const BYTE val,
            const BYTE 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

4.8.2.9 get_fp_dec()

```
int32_t get_fp_dec (
            float f,
            uint8_t nb )  [inline]
```

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, so welcome get_fp_dec for the purpose.

Parameters

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

Returns

nb decimal part as integer

4.8.2.10 gray2bin()

```
DWORD gray2bin (
            const DWORD gray )  [inline]
```

Convert gray code to binary value.

Parameters

in	*gray*	- gray code value

Returns

Converted value (binary)

4.8.2.11 HexToASCII()

```
CHAR INLINE__ HexToASCII (
            BYTE hex )  [inline]
```

Converts hexadecimal value to ASCII.

Parameters

in	*hex*	- Hexadecimal value to convert

Returns

ASCII char

4.8.2.12 HexToBCD()

```
BYTE HexToBCD (
            const BYTE hex )  [inline]
```

Converts hexadecimal value to BCD.

Note

Returns 0xFF if Hex value can't be represented on a BCD BYTE

Parameters

in	*hex*	- Hexadecimal value to convert

Returns

BCD value

4.8.2.13 inRange()

```
bool INLINE__ inRange (
            const SDWORD val,
            const SDWORD low,
            const SDWORD 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

4.8.2.14 inTolerance()

```
bool inTolerance (
             const SDWORD val,
             const SDWORD ref,
             float 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

4.8.2.15 SWAP_END16B()

```
WORD SWAP_END16B (
             const WORD w )  [inline]
```

Swap endians of the contents of a 16b value (little -> big, big -> little)

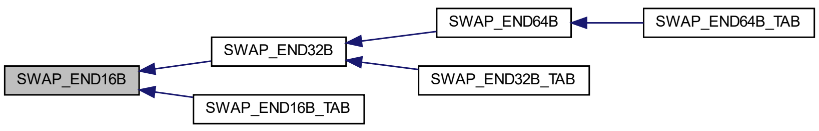
Parameters

in	*w*	- 16b value

Returns

Swapped value

Here is the caller graph for this function:



4.8.2.16 SWAP_END16B_TAB()

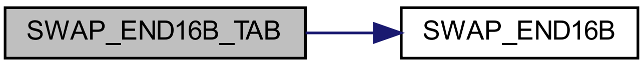
```
void INLINE__ SWAP_END16B_TAB (
            WORD tab[],
            const WORD nb )  [inline]
```

Swap endians of a 16b tab (little -> big, big -> little)

Parameters

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

Here is the call graph for this function:



4.8.2.17 SWAP_END32B()

```
DWORD SWAP_END32B (
            const DWORD d )  [inline]
```

Swap endians of the contents of a 32b value (little -> big, big -> little)

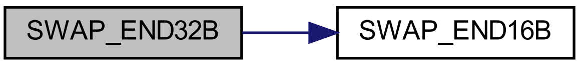
Parameters

in	*d*	- 32b value

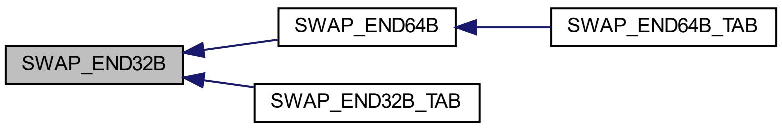
Returns

Swapped value

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.18 SWAP_END32B_TAB()

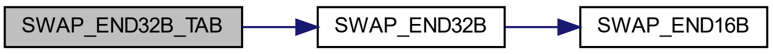
```
void INLINE__ SWAP_END32B_TAB (
             DWORD tab[],
             const WORD nb )  [inline]
```

Swap endians of a 32b tab (little -> big, big -> little)

Parameters

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

Here is the call graph for this function:



4.8.2.19 SWAP_END64B()

```
LWORD SWAP_END64B (
             const LWORD l )  [inline]
```

Swap endians of the contents of a 64b value (little -> big, big -> little)

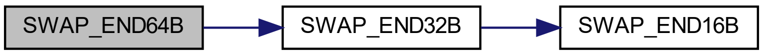
Parameters

in	*l*	- 64b value

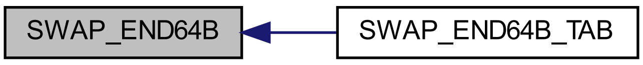
Returns

Swapped value

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.20 SWAP_END64B_TAB()

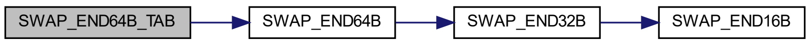
```
void INLINE__ SWAP_END64B_TAB (
             LWORD tab[],
             const WORD nb )  [inline]
```

Swap endians of a 64b tab (little -> big, big -> little)

Parameters

in	*tab*	- tab of 64b values
in	*nb*	- nb of values in tab

Here is the call graph for this function:



4.8.2.21 testEndian_basic()

```
eEndian testEndian_basic (
            void )  [inline]
```

Test Core endian.

Returns

Endian type

4.8.2.22 testEndian_full()

```
eEndian testEndian_full (
            void )  [inline]
```

Test Core endian (full, recognizing mid endians too)

Returns

Endian type

4.8.2.23 TPSINF_MS()

```
bool INLINE__ TPSINF_MS (
            const DWORD last,
            const DWORD time )  [inline]
```

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

Warning

For SAM families, no ms base time counter is implemented in HAL, please refer to arm_chip_sam.h for an implementation example.

Note

Define custom HAL_MS_TICKS_FACTOR at project level if tick period is not 1ms

Parameters

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

Returns

true if time not elapsed

4.8.2.24 TPSSUP_MS()

```
bool INLINE__ TPSSUP_MS (
            const DWORD last,
            const DWORD time )  [inline]
```

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

Warning

For SAM families, no ms base time counter is implemented in HAL, please refer to arm_chip_sam.h for an implementation example.

Note

Define custom HAL_MS_TICKS_FACTOR at project level if tick period is not 1ms

Parameters

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

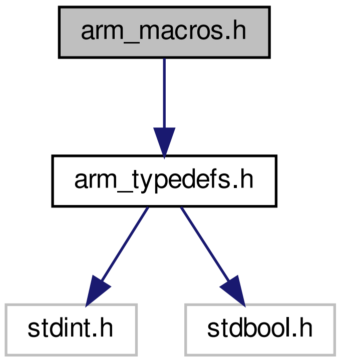
Returns

true if time elapsed

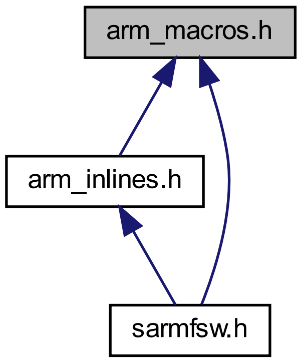
4.9 arm_macros.h File Reference

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 True true

  True** alias for **true
- #define False false

  False** alias for **false
- #define TRUE true

  TRUE** alias for **true
- #define FALSE false

  FALSE** alias for **false
- #define LSHIFT(v, b) ((v) ∗ (1UL << b))
- #define RSHIFT(v, b) ((v) / (1UL << b))
- #define LSHIFT64(v, b) ((v) ∗ (1ULL << b))
- #define RSHIFT64(v, b) ((v) / (1ULL << b))

  *Shift **v b** bits right (up to 63b)*
- #define MAKEWORD(lsb, msb) ((WORD) (((BYTE) (lsb)) | LSHIFT(((WORD) ((BYTE) (msb))), 8)))

  *Make WORD from **lsb** and **msb**.*
- #define MAKELONG(lsw, msw) ((DWORD) (((WORD) (lsw)) | LSHIFT(((DWORD) ((WORD) (msw))), 16)))

  *Make LONG from **lsw** and **msw**.*
- #define LOWORD(l) ((WORD) (l))

  *Get WORD LSW from LONG **l**.*
- #define HIWORD(l) ((WORD) RSHIFT((DWORD) (l), 16))

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

  *Get BYTE LSB from WORD **w**.*
- #define HIBYTE(w) ((BYTE) RSHIFT((WORD) (w), 8))

  *Get BYTE MSB from WORD **w**.*
- #define SWAP_TYPE(a, b, typ) { typ c = a; a = b; b = c; }

  *Swap type **typ** **a** & **b**.*
- #define SWAP_BYTE(a, b) SWAP_TYPE(a, b, BYTE)

  *Swap BYTEs **a** & **b**.*
- #define SWAP_WORD(a, b) SWAP_TYPE(a, b, WORD)

  *Swap WORDs **a** & **b**.*
- #define SWAP_DWORD(a, b) SWAP_TYPE(a, b, DWORD)

  *Swap DWORDs **a** & **b**.*
- #define SWAP_LWORD(a, b) SWAP_TYPE(a, b, LWORD)

  *Swap LWORDs **a** & **b**.*
- #define SWAP_FLOAT(a, b) SWAP_TYPE(a, b, float)

  *Swap floats **a** & **b**.*
- #define SWAP_DOUBLE(a, b) SWAP_TYPE(a, b, double)

  *Swap doubles **a** & **b**.*
- #define SZ_OBJ(obj, typ) ((size_t) (sizeof(obj) / sizeof(typ)))

  *Computes the number of elements of **obj** following **typ**.*
- #define OFFSET_OF(typ, mbr) ((size_t) &(((typ ∗)0)->mbr))

  *Computes the offset member **mbr** from struct **typ**.*
- #define ROOT_OF(ptr, typ, mbr) ((typ ∗) (((uint8_t ∗) ptr) - OFFSET_OF(typ, mbr)))

  *Computes the address of parent struct **typ** of **ptr** from member **mbr**.*
- #define CAT(a, b) a##b

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

  Preprocessor Name catenation (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 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 RAD_TO_FLOAT(r) ((float) (((r) > 2∗Pi ? 2∗Pi : (r)) / 2∗Pi))
- #define FLOAT_TO_RAD(f) ((float) ((((f) > 1.0f ? 1.0f : (f)) < 0.0f ? 0.0f : (f)) ∗ 2∗Pi)
- #define DEG_TO_FLOAT(d) ((float) (((d) > 360.0f ? 360.0f : (d)) / 360.0f))
- #define FLOAT_TO_DEG(f) ((float) ((((f) > 1.0f ? 1.0f : (f)) < 0.0f ? 0.0f : (f)) ∗ 360.0f))

4.9.1 Detailed Description

ARM common macros.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.9.2 Macro Definition Documentation

4.9.2.1 binEval

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

boolean evaluation of expression **exp**

4.9.2.2 BYTE_TO_PERC

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

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

4.9.2.3 CAT

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

Preprocessor Name catenation.

Warning

No nesting possible, use *XCAT* in this case

4.9.2.4 charNUL

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

Null Char.

4.9.2.5 CLAMP

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

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

4.9.2.6 DEG_TO_FLOAT

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

4.9.2.7 False

```
#define False false
```

False alias for **false**

4.9.2.8 FALSE

```
#define FALSE false
```

FALSE alias for **false**

4.9.2.9 FLOAT_TO_DEG

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

4.9.2.10 FLOAT_TO_RAD

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

4.9.2.11 HIBYTE

```
#define HIBYTE(
              w ) ((BYTE) RSHIFT((WORD) (w), 8))
```

Get BYTE MSB from WORD **w**.

4.9.2.12 HIWORD

```
#define HIWORD(
              l ) ((WORD) RSHIFT((DWORD) (l), 16))
```

Get WORD MSW from LONG **l**.

4.9.2.13 LOBYTE

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

Get BYTE LSB from WORD **w**.

4.9.2.14 LOWORD

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

Get WORD LSW from LONG **l**.

4.9.2.15 LSHIFT

```
#define LSHIFT(
            v,
            b ) ((v) * (1UL << b))
```

Warning

this macro is optimized only when used with **b** with a static value Shift **v** **b** bits left (up to 31b)

4.9.2.16 LSHIFT64

```
#define LSHIFT64(
            v,
            b ) ((v) * (1ULL << b))
```

Warning

this macro is optimized only when used with **b** with a static value Shift **v** **b** bits left (up to 63b)

4.9.2.17 MAKELONG

```
#define MAKELONG(
            lsw,
            msw ) ((DWORD) (((WORD) (lsw)) | LSHIFT(((DWORD) ((WORD) (msw))), 16)))
```

Make LONG from **lsw** and **msw**.

4.9.2.18 MAKEWORD

```
#define MAKEWORD(
            lsb,
            msb ) ((WORD) (((BYTE) (lsb)) | LSHIFT(((WORD) ((BYTE) (msb))), 8)))
```

Make WORD from **lsb** and **msb**.

4.9.2.19 max

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

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

4.9.2.20 MAX3

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

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

4.9.2.21 min

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

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

4.9.2.22 MIN3

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

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

4.9.2.23 nbinEval

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

complemented boolean evaluation of expression **exp**

4.9.2.24 Null

```
#define Null 0
```

Null Value.

4.9.2.25 OFFSET_OF

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

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

4.9.2.26 OneThird

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

1/3 approximation

4.9.2.27 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.

4.9.2.28 Pi

```
#define Pi 3.141593f
```

Approximate Pi calculation (4 ∗ atan(1))

4.9.2.29 pNull

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

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

4.9.2.30 RAD_TO_FLOAT

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

4.9.2.31 ROOT_OF

```
#define ROOT_OF(
              ptr,
              typ,
              mbr ) ((typ *) (((uint8_t *) ptr) - OFFSET_OF(typ, mbr)))
```

Computes the address of parent struct **typ** of **ptr** from member **mbr**.

4.9.2.32 RSHIFT

```
#define RSHIFT(
              v,
              b ) ((v) / (1UL << b))
```

Warning

this macro is optimized only when used with **b** with a static value Shift **v** **b** bits right (up to 31b)

4.9.2.33 RSHIFT64

```
#define RSHIFT64(
              v,
              b ) ((v) / (1ULL << b))
```

Shift **v** **b** bits right (up to 63b)

Warning

this macro is optimized only when used with **b** with a static value

4.9.2.34 STR

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

Stringify an expression.

4.9.2.35 SWAP_BYTE

```
#define SWAP_BYTE(
              a,
              b ) SWAP_TYPE(a, b, BYTE)
```

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

4.9.2.36 SWAP_DOUBLE

```
#define SWAP_DOUBLE(
              a,
              b ) SWAP_TYPE(a, b, double)
```

Swap doubles **a** & **b**.

4.9.2.37 SWAP_DWORD

```
#define SWAP_DWORD(
              a,
              b ) SWAP_TYPE(a, b, DWORD)
```

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

4.9.2.38 SWAP_FLOAT

```
#define SWAP_FLOAT(
              a,
              b ) SWAP_TYPE(a, b, float)
```

Swap floats **a** & **b**.

4.9.2.39 SWAP_LWORD

```
#define SWAP_LWORD(
              a,
              b ) SWAP_TYPE(a, b, LWORD)
```

Swap LWORDs **a** & **b**.

4.9.2.40 SWAP_TYPE

```
#define SWAP_TYPE(
              a,
              b,
              typ ) { typ c = a; a = b; b = c; }
```

Swap type **typ** **a** & **b**.

4.9.2.41 SWAP_WORD

```
#define SWAP_WORD(
              a,
              b ) SWAP_TYPE(a, b, WORD)
```

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

4.9.2.42 SZ_OBJ

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

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

4.9.2.43 True

```
#define True true
```

True alias for **true**

4.9.2.44 TRUE

```
#define TRUE true
```

TRUE alias for **true**

4.9.2.45 TwoThird

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

2/3 approximation

4.9.2.46 Undefined

```
#define Undefined -1
```

Undefined value.

4.9.2.47 XCAT

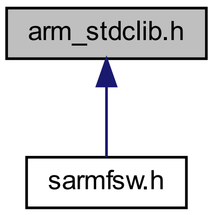
```
#define XCAT(
              a,
              b ) CAT(a, b)
```

Preprocessor Name catenation (possible nesting)

4.10 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.

4.10.1 Detailed Description

ARM common standard c library wrapper macros.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

4.10.2 Macro Definition Documentation

4.10.2.1 printExpr

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

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

4.10.2.2 str_add_cr

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

Adding new line to string using strcat.

4.10.2.3 str_add_tab

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

Adding tab to string using strcat.

4.10.2.4 str_clr

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

clear string **s** (fast way)

4.10.2.5 str_clr_safe

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

clear string **s** (safe way)

4.10.2.6 TestMalloc

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

Asserted malloc.

4.10.2.7 verbInstr

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

Print instruction **e** and execute it.

4.10.2.8 VerboseInc

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

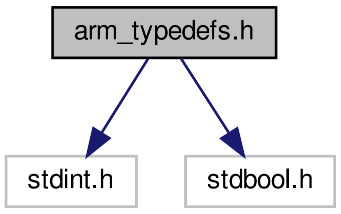
Increment example using puts.

4.11 arm_typedefs.h File Reference

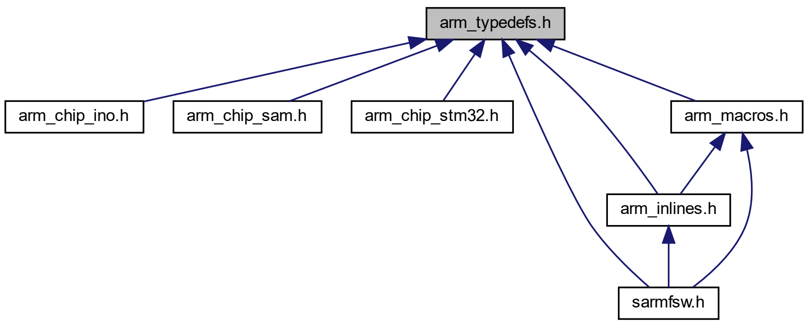
ARM common typedefs.

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

Include dependency graph for arm_typedefs.h:



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



Data Structures

- 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 enum eGPIOState eGPIOState
- typedef enum eGPIOPull eGPIOPull
- typedef enum eEndian eEndian
- 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.
- enum eGPIOState { Reset = 0, Set, Toggle }

  GPIO possible states/actions enumeration.
- enum eGPIOPull { PullDown = 0, PullUp, NoPull }
- enum eEndian {
  Endian_little = 0, Endian_big, Endian_mid_little, Endian_mid_big,
  Endian_unknown }

  Core endian.

4.11.1 Detailed Description

ARM common typedefs.

Author

SMFSW

Copyright

MIT (c) 2017-2018, SMFSW

Warning

Endianness for unions shall be checked following target / compiler to avoid potential headaches
sBitfieldXX are defined from lsb to msb as most compiler does by default; if it's not the case try to find a compiler directive or pragma to reverse bitfield order. If not available, define REVERSE_BITFIELD symbol at project level.
For Arduino platform, some binary.h definitions needs to be undefined, If you find them useful, define I_F↩
IND_BINARY_HEADER_USEFUL in project to redefine them Please note, B0 & B1 Bytes sub-structures of sBitfieldXX will not be available anymore

4.11.2 Typedef Documentation

4.11.2.1 BYTE

```
typedef uint8_t BYTE
```

Unsigned Byte typedef (8bits)

4.11.2.2 CHAR

```
typedef char CHAR
```

Char typedef (8bits)

4.11.2.3 DWORD

```
typedef uint32_t DWORD
```

Unsigned dWord typedef (32bits)

4.11.2.4 eEdge

```
typedef enum eEdge eEdge
```

4.11.2.5 eEndian

```
typedef enum eEndian eEndian
```

4.11.2.6 eGPIOPull

```
typedef enum eGPIOPull eGPIOPull
```

4.11.2.7 eGPIOState

```
typedef enum eGPIOState eGPIOState
```

4.11.2.8 eState

```
typedef enum eState eState
```

4.11.2.9 LWORD

```
typedef uint64_t LWORD
```

Unsigned lWord typedef (64bits)

4.11.2.10 sBitfield16

```
typedef struct StructBitfield16 sBitfield16
```

4.11.2.11 sBitfield32

```
typedef struct StructBitfield32 sBitfield32
```

4.11.2.12 sBitfield64

```
typedef struct StructBitfield64 sBitfield64
```

4.11.2.13 sBitfield8

```
typedef struct StructBitfield8 sBitfield8
```

4.11.2.14 SBYTE

```
typedef int8_t SBYTE
```

Signed Byte typedef (8bits)

4.11.2.15 SDWORD

```
typedef int32_t SDWORD
```

Signed dWord typedef (32bits)

4.11.2.16 SLWORD

```
typedef int64_t SLWORD
```

Signed lWord typedef (64bits)

4.11.2.17 SWORD

```
typedef int16_t SWORD
```

Signed Word typedef (16bits)

4.11.2.18 uByte

```
typedef union UnionByte uByte
```

4.11.2.19 uDWord

```
typedef union UnionDWord uDWord
```

4.11.2.20 uLWord

```
typedef union UnionLWord uLWord
```

4.11.2.21 uWord

```
typedef union UnionWord uWord
```

4.11.2.22 WORD

```
typedef uint16_t WORD
```

Unsigned Word typedef (16bits)

4.11.3 Enumeration Type Documentation

4.11.3.1 eEdge

```
enum eEdge
```

Signal Edges.

Enumerator

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

4.11.3.2 eEndian

```
enum eEndian
```

Core endian.

Enumerator

Endian_little	Little endian configured MCU.
Endian_big	Big endian configured MCU.
Endian_mid_little	Middle little endian configured MCU (PDP-11)
Endian_mid_big	Middle big endian configured MCU (Honeywell 316)
Endian_unknown	Unknown endian MCU.

4.11.3.3 eGPIOPull

```
enum eGPIOPull
```

Enumerator

PullDown	GPIO with pull down.
PullUp	GPIO with pull up.
NoPull	GPIO without pull.

4.11.3.4 eGPIOState

```
enum eGPIOState
```

GPIO possible states/actions enumeration.

Enumerator

Reset	Reset State.
Set	Set State.
Toggle	Toggle Output **Note** Toggle is only GPIO output related

4.11.3.5 eState

```
enum eState
```

Activation state On, Off.

Enumerator

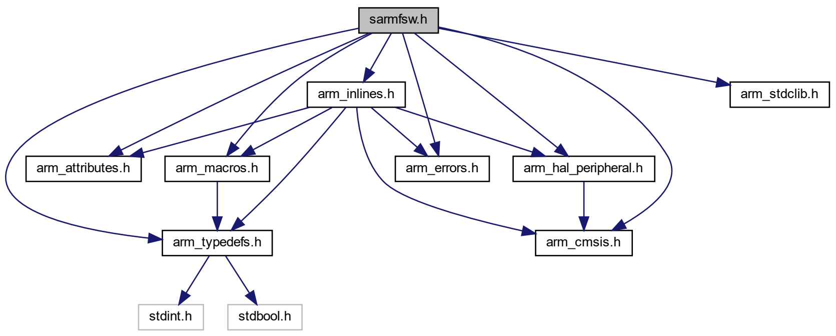
Off	Off / Reset.
On	On / Set.

4.12 sarmfsw.h File Reference

sarmfsw ARM common headers

```
#include "arm_attributes.h"
#include "arm_typedefs.h"
#include "arm_errors.h"
#include "arm_macros.h"
#include "arm_stdclib.h"
#include "arm_cmsis.h"
#include "arm_hal_peripheral.h"
#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.

4.12.1 Detailed Description

sarmfsw ARM common headers

Author

SMFSW

Copyright

MIT (c) 2017-208, SMFSW

4.12.2 Typedef Documentation

4.12.2.1 FW_target

```
typedef enum FW_target FW_target
```

4.12.3 Enumeration Type Documentation

4.12.3.1 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.

Index