

MSP430-LEI

Generated by Doxygen 1.7.6.1

Wed Jan 23 2013 16:02:21

Contents

1	Module Index	1
1.1	Modules	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	Personal Errors	7
4.1.1	Define Documentation	13
4.1.1.1	E2BIG	13
4.1.1.2	EACCES	13
4.1.1.3	EADDRINUSE	14
4.1.1.4	EADDRNOTAVAIL	14
4.1.1.5	EADV	14
4.1.1.6	EAFNOSUPPORT	14
4.1.1.7	EAGAIN	14
4.1.1.8	EALREADY	14
4.1.1.9	EBADE	14
4.1.1.10	EBADF	14
4.1.1.11	EBADFD	15
4.1.1.12	EBADMSG	15
4.1.1.13	EBADR	15
4.1.1.14	EBADRQC	15

4.1.1.15	EBADSLT	15
4.1.1.16	EBFONT	15
4.1.1.17	EBUSY	15
4.1.1.18	ECANCELED	15
4.1.1.19	ECHILD	16
4.1.1.20	ECHRNG	16
4.1.1.21	ECOMM	16
4.1.1.22	ECONNABORTED	16
4.1.1.23	ECONNREFUSED	16
4.1.1.24	ECONNRESET	16
4.1.1.25	EDEADLK	16
4.1.1.26	EDEADLOCK	16
4.1.1.27	EDESTADDRREQ	17
4.1.1.28	EDOM	17
4.1.1.29	EDOTDOT	17
4.1.1.30	EDQUOT	17
4.1.1.31	EEXIST	17
4.1.1.32	EFAULT	17
4.1.1.33	EFBIG	17
4.1.1.34	EHOSTDOWN	17
4.1.1.35	EHOSTUNREACH	18
4.1.1.36	EHWPOISON	18
4.1.1.37	EIDRM	18
4.1.1.38	EILSEQ	18
4.1.1.39	EINPROGRESS	18
4.1.1.40	EINTR	18
4.1.1.41	EINVAL	18
4.1.1.42	EIO	18
4.1.1.43	EISCONN	19
4.1.1.44	EISDIR	19
4.1.1.45	EISNAM	19
4.1.1.46	EKEYEXPIRED	19
4.1.1.47	EKEYREJECTED	19
4.1.1.48	EKEYREVOKED	19

4.1.1.49	EL2HLT	19
4.1.1.50	EL2NSYNC	19
4.1.1.51	EL3HLT	20
4.1.1.52	EL3RST	20
4.1.1.53	ELIBACC	20
4.1.1.54	ELIBBAD	20
4.1.1.55	ELIBEXEC	20
4.1.1.56	ELIBMAX	20
4.1.1.57	ELIBSCN	20
4.1.1.58	ELNRNG	20
4.1.1.59	ELOOP	21
4.1.1.60	EMEDIUMTYPE	21
4.1.1.61	EMFILE	21
4.1.1.62	EMLINK	21
4.1.1.63	EMSGSIZE	21
4.1.1.64	EMULTIHOP	21
4.1.1.65	ENAMETOOLONG	21
4.1.1.66	ENAVAIL	21
4.1.1.67	ENETDOWN	22
4.1.1.68	ENETRESET	22
4.1.1.69	ENETUNREACH	22
4.1.1.70	ENFILE	22
4.1.1.71	ENOANO	22
4.1.1.72	ENOBUFFS	22
4.1.1.73	ENOCSS	22
4.1.1.74	ENODATA	22
4.1.1.75	ENODEV	23
4.1.1.76	ENOENT	23
4.1.1.77	ENOEXEC	23
4.1.1.78	ENOKEY	23
4.1.1.79	ENOLCK	23
4.1.1.80	ENOLINK	23
4.1.1.81	ENOMEDIUM	23
4.1.1.82	ENOMEM	23

4.1.1.83	ENOMSG	24
4.1.1.84	ENONET	24
4.1.1.85	ENOPKG	24
4.1.1.86	ENOPROTOOPT	24
4.1.1.87	ENOSPC	24
4.1.1.88	ENOSR	24
4.1.1.89	ENOSTR	24
4.1.1.90	ENOSYS	24
4.1.1.91	ENOTBLK	25
4.1.1.92	ENOTCONN	25
4.1.1.93	ENOTDIR	25
4.1.1.94	ENOTEMPTY	25
4.1.1.95	ENOTNAM	25
4.1.1.96	ENOTRECOVERABLE	25
4.1.1.97	ENOTSOCK	25
4.1.1.98	ENOTTY	25
4.1.1.99	ENOTUNIQ	26
4.1.1.100	ENXIO	26
4.1.1.101	EOPNOTSUPP	26
4.1.1.102	EOVERFLOW	26
4.1.1.103	EOWNERDEAD	26
4.1.1.104	EPERM	26
4.1.1.105	EPFNOSUPPORT	26
4.1.1.106	EPIPE	26
4.1.1.107	EPROTO	27
4.1.1.108	EPROTONOSUPPORT	27
4.1.1.109	EPROTOTYPE	27
4.1.1.110	ERANGE	27
4.1.1.111	EREMCHG	27
4.1.1.112	EREMOTE	27
4.1.1.113	EREMOTEIO	27
4.1.1.114	ERESTART	27
4.1.1.115	ERFKILL	28
4.1.1.116	EROFS	28

4.1.1.117 ESHUTDOWN	28
4.1.1.118 ESOCKTNOSUPPORT	28
4.1.1.119 ESPIPE	28
4.1.1.120 ESRCH	28
4.1.1.121 ESRMNT	28
4.1.1.122 ESTALE	28
4.1.1.123 ESTRPIPE	29
4.1.1.124 ETIME	29
4.1.1.125 ETIMEDOUT	29
4.1.1.126 ETOOMANYREFS	29
4.1.1.127 ETXTBSY	29
4.1.1.128 EUCLEAN	29
4.1.1.129 EUNATCH	29
4.1.1.130 EUSERS	29
4.1.1.131 EWOULDBLOCK	30
4.1.1.132 EXDEV	30
4.1.1.133 EXFULL	30
4.1.1.134 SUCCESS	30
 5 Data Structure Documentation	 31
5.1 DIR Struct Reference	31
5.1.1 Detailed Description	31
5.1.2 Field Documentation	31
5.1.2.1 clust	31
5.1.2.2 fn	31
5.1.2.3 index	32
5.1.2.4 sclust	32
5.1.2.5 sect	32
5.2 FATFS Struct Reference	32
5.2.1 Detailed Description	32
5.2.2 Field Documentation	33
5.2.2.1 csize	33
5.2.2.2 curr_clust	33
5.2.2.3 database	33

5.2.2.4	dirbase	33
5.2.2.5	dsect	33
5.2.2.6	fatbase	33
5.2.2.7	flag	33
5.2.2.8	fptr	33
5.2.2.9	fs_type	33
5.2.2.10	fsize	33
5.2.2.11	n_fatent	34
5.2.2.12	n_rootdir	34
5.2.2.13	org_clust	34
5.2.2.14	pad1	34
5.3	FILINFO Struct Reference	34
5.3.1	Detailed Description	34
5.3.2	Field Documentation	34
5.3.2.1	fattrib	34
5.3.2.2	fdate	35
5.3.2.3	fname	35
5.3.2.4	fsize	35
5.3.2.5	ftime	35
6	File Documentation	37
6.1	gps/gps.c File Reference	37
6.1.1	Function Documentation	38
6.1.1.1	gps_initialize	38
6.1.1.2	gps_read	38
6.2	gps/gps.h File Reference	38
6.2.1	Function Documentation	39
6.2.1.1	gps_initialize	39
6.2.1.2	gps_read	39
6.3	mmc/diskio.h File Reference	40
6.3.1	Define Documentation	41
6.3.1.1	_DISKIO	41
6.3.1.2	CT_BLOCK	41
6.3.1.3	CT_MMC	41

6.3.1.4	CT_SD1	41
6.3.1.5	CT_SD2	41
6.3.1.6	CT_SDC	42
6.3.1.7	STA_NODISK	42
6.3.1.8	STA_NOINIT	42
6.3.2	Typedef Documentation	42
6.3.2.1	DSTATUS	42
6.3.3	Enumeration Type Documentation	42
6.3.3.1	DRESULT	42
6.3.4	Function Documentation	42
6.3.4.1	disk_initialize	42
6.3.4.2	disk_readp	43
6.3.4.3	disk_writep	43
6.4	mmc/integer.h File Reference	43
6.4.1	Typedef Documentation	44
6.4.1.1	BYTE	44
6.4.1.2	CHAR	44
6.4.1.3	DWORD	44
6.4.1.4	INT	44
6.4.1.5	LONG	44
6.4.1.6	SHORT	44
6.4.1.7	UCHAR	44
6.4.1.8	UINT	44
6.4.1.9	ULONG	44
6.4.1.10	USHORT	45
6.4.1.11	WCHAR	45
6.4.1.12	WORD	45
6.5	mmc/mmc.c File Reference	45
6.5.1	Define Documentation	46
6.5.1.1	ACMD41	46
6.5.1.2	CMD0	46
6.5.1.3	CMD1	46
6.5.1.4	CMD16	46
6.5.1.5	CMD17	46

6.5.1.6	CMD24	47
6.5.1.7	CMD55	47
6.5.1.8	CMD58	47
6.5.1.9	CMD8	47
6.5.1.10	CT_BLOCK	47
6.5.1.11	CT_MMC	47
6.5.1.12	CT_SD1	47
6.5.1.13	CT_SD2	47
6.5.1.14	DELAY_100US	48
6.5.1.15	DESELECT	48
6.5.1.16	FORWARD	48
6.5.1.17	MMC_SEL	48
6.5.1.18	SELECT	48
6.5.2	Function Documentation	48
6.5.2.1	disk_initialize	48
6.5.2.2	disk_readp	48
6.5.2.3	disk_writep	48
6.6	mmc/pff.c File Reference	49
6.6.1	Define Documentation	50
6.6.1.1	_DF1S	50
6.6.1.2	BPB_BkBootSec	51
6.6.1.3	BPB_BytsPerSec	51
6.6.1.4	BPB_ExtFlags	51
6.6.1.5	BPB_FATSz16	51
6.6.1.6	BPB_FATSz32	51
6.6.1.7	BPB_FSInfo	51
6.6.1.8	BPB_FSVer	51
6.6.1.9	BPB_HiddSec	51
6.6.1.10	BPB_Media	51
6.6.1.11	BPB_NumFATs	51
6.6.1.12	BPB_NumHeads	52
6.6.1.13	BPB_RootClus	52
6.6.1.14	BPB_RootEntCnt	52
6.6.1.15	BPB_RsvdSecCnt	52

6.6.1.16	BPB_SecPerClus	52
6.6.1.17	BPB_SecPerTrk	52
6.6.1.18	BPB_TotSec16	52
6.6.1.19	BPB_TotSec32	52
6.6.1.20	BS_55AA	52
6.6.1.21	BS_BootSig	52
6.6.1.22	BS_BootSig32	53
6.6.1.23	BS_DrvNum	53
6.6.1.24	BS_DrvNum32	53
6.6.1.25	BS_FilSysType	53
6.6.1.26	BS_FilSysType32	53
6.6.1.27	BS_jmpBoot	53
6.6.1.28	BS_OEMName	53
6.6.1.29	BS_VolID	53
6.6.1.30	BS_VolID32	53
6.6.1.31	BS_VolLab	53
6.6.1.32	BS_VolLab32	54
6.6.1.33	DIR_Attr	54
6.6.1.34	DIR_CrtDate	54
6.6.1.35	DIR_CrtTime	54
6.6.1.36	DIR_FileSize	54
6.6.1.37	DIR_FstClusHI	54
6.6.1.38	DIR_FstClusLO	54
6.6.1.39	DIR_Name	54
6.6.1.40	DIR_NTres	54
6.6.1.41	DIR_WrtDate	54
6.6.1.42	DIR_WrtTime	55
6.6.1.43	IsDBCS1	55
6.6.1.44	IsDBCS2	55
6.6.1.45	IsLower	55
6.6.1.46	IsUpper	55
6.6.1.47	LD_CLUST	55
6.6.1.48	MBR_Table	55
6.6.2	Function Documentation	55

6.6.2.1	pf_lseek	55
6.6.2.2	pf_mount	55
6.6.2.3	pf_open	55
6.6.2.4	pf_opendir	56
6.6.2.5	pf_read	56
6.6.2.6	pf_readdir	56
6.6.2.7	pf_write	56
6.7	mmc/pff.h File Reference	56
6.7.1	Define Documentation	58
6.7.1.1	_CODE_PAGE	58
6.7.1.2	_FS_FAT12	58
6.7.1.3	_FS_FAT32	58
6.7.1.4	_USE_DIR	59
6.7.1.5	_USE_LSEEK	59
6.7.1.6	_USE_READ	59
6.7.1.7	_USE_WRITE	59
6.7.1.8	_WORD_ACCESS	59
6.7.1.9	AM_ARC	59
6.7.1.10	AM_DIR	59
6.7.1.11	AM_HID	59
6.7.1.12	AM_LFN	59
6.7.1.13	AM_MASK	59
6.7.1.14	AM_RDO	60
6.7.1.15	AM_SYS	60
6.7.1.16	AM_VOL	60
6.7.1.17	CLUST	60
6.7.1.18	FA__WIP	60
6.7.1.19	FA_OPENED	60
6.7.1.20	FA_WPRT	60
6.7.1.21	FS_FAT12	60
6.7.1.22	FS_FAT16	60
6.7.1.23	FS_FAT32	60
6.7.1.24	LD_DWORD	61
6.7.1.25	LD_WORD	61

6.7.1.26	ST_DWORD	61
6.7.1.27	ST_WORD	61
6.7.2	Enumeration Type Documentation	61
6.7.2.1	FRESULT	61
6.7.3	Function Documentation	61
6.7.3.1	pf_lseek	61
6.7.3.2	pf_mount	62
6.7.3.3	pf_open	62
6.7.3.4	pf_opendir	62
6.7.3.5	pf_read	62
6.7.3.6	pf_readdir	62
6.7.3.7	pf_write	62
6.8	myerrno.h File Reference	63
6.9	spi/spi.c File Reference	70
6.9.1	Define Documentation	71
6.9.1.1	HIBYTE	71
6.9.1.2	LOBYTE	71
6.9.1.3	SPI_MODE_0	71
6.9.1.4	SPI_MODE_1	71
6.9.1.5	SPI_MODE_2	71
6.9.1.6	SPI_MODE_3	71
6.9.2	Function Documentation	71
6.9.2.1	spi_initialize	71
6.9.2.2	spi_receive	72
6.9.2.3	spi_send	72
6.9.2.4	spi_set_divisor	72
6.10	spi/spi.h File Reference	72
6.10.1	Define Documentation	73
6.10.1.1	SPI_16MHz	73
6.10.1.2	SPI_1MHz	73
6.10.1.3	SPI_250kHz	74
6.10.1.4	SPI_2MHz	74
6.10.1.5	SPI_400kHz	74
6.10.1.6	SPI_4MHz	74

6.10.1.7	SPI_8MHz	74
6.10.1.8	SPI_DEFAULT_SPEED	74
6.10.2	Function Documentation	74
6.10.2.1	spi_initialize	74
6.10.2.2	spi_receive	74
6.10.2.3	spi_send	75
6.10.2.4	spi_set_divisor	75
6.11	uart/uart.c File Reference	75
6.11.1	Function Documentation	76
6.11.1.1	getchar	76
6.11.1.2	interrupt	76
6.11.1.3	putchar	76
6.11.1.4	uart_close	76
6.11.1.5	uart_initialize	76
6.11.1.6	uart_set_bitrate	77
6.12	uart/uart.h File Reference	77
6.12.1	Function Documentation	78
6.12.1.1	getchar	78
6.12.1.2	putchar	78
6.12.1.3	uart_close	78
6.12.1.4	uart_initialize	78

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Personal Errors	7
---------------------------	---

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

DIR	31
FATFS	32
FILINFO	34

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

myerrno.h	63
gps/gps.c	37
gps/gps.h	38
mmc/diskio.h	40
mmc/integer.h	43
mmc/mmc.c	45
mmc/pff.c	49
mmc/pff.h	56
spi/spi.c	70
spi/spi.h	72
uart/uart.c	75
uart/uart.h	77

Chapter 4

Module Documentation

4.1 Personal Errors

Defines

- #define `SUCCESS` 0
No Error.
- #define `EPERM` 1
Operation not permitted.
- #define `ENOENT` 2
No such file or directory.
- #define `ESRCH` 3
No such process.
- #define `EINTR` 4
Interrupted system call.
- #define `EIO` 5
I/O error.
- #define `ENXIO` 6
No such device or address.
- #define `E2BIG` 7
Argument list too long.
- #define `ENOEXEC` 8
Exec format error.
- #define `EBADF` 9
Bad file number.
- #define `ECHILD` 10
No child processes.
- #define `EAGAIN` 11
Try again.
- #define `ENOMEM` 12

- Out of memory.*
- #define [EACCES](#) 13
- Permission denied.*
- #define [EFAULT](#) 14
- Bad address.*
- #define [ENOTBLK](#) 15
- Block device required.*
- #define [EBUSY](#) 16
- Device or resource busy.*
- #define [EEXIST](#) 17
- File exists.*
- #define [EXDEV](#) 18
- Cross-device link.*
- #define [ENODEV](#) 19
- No such device.*
- #define [ENOTDIR](#) 20
- Not a directory.*
- #define [EISDIR](#) 21
- Is a directory.*
- #define [EINVAL](#) 22
- Invalid argument.*
- #define [ENFILE](#) 23
- File table overflow.*
- #define [EMFILE](#) 24
- Too many open files.*
- #define [ENOTTY](#) 25
- Not a typewriter.*
- #define [ETXTBSY](#) 26
- Text file busy.*
- #define [EFBIG](#) 27
- File too large.*
- #define [ENOSPC](#) 28
- No space left on device.*
- #define [ESPIPE](#) 29
- Illegal seek.*
- #define [EROFS](#) 30
- Read-only file system.*
- #define [EMLINK](#) 31
- Too many links.*
- #define [EPIPE](#) 32
- Broken pipe.*
- #define [EDOM](#) 33
- Math argument out of domain of func.*

- #define [ERANGE](#) 34
Math result not representable.
- #define [EDEADLK](#) 35
Resource deadlock would occur.
- #define [ENAMETOOLONG](#) 36
File name too long.
- #define [ENOLCK](#) 37
No record locks available.
- #define [ENOSYS](#) 38
Function not implemented.
- #define [ENOTEMPTY](#) 39
Directory not empty.
- #define [ELOOP](#) 40
Too many symbolic links encountered.
- #define [EWOULDBLOCK](#) [EAGAIN](#)
Operation would block.
- #define [ENOMSG](#) 42
No message of desired type.
- #define [EIDRM](#) 43
Identifier removed.
- #define [ECHRNG](#) 44
Channel number out of range.
- #define [EL2NSYNC](#) 45
Level 2 not synchronized.
- #define [EL3HLT](#) 46
Level 3 halted.
- #define [EL3RST](#) 47
Level 3 reset.
- #define [ELNRNG](#) 48
Link number out of range.
- #define [EUNATCH](#) 49
Protocol driver not attached.
- #define [ENOCSI](#) 50
No CSI structure available.
- #define [EL2HLT](#) 51
Level 2 halted.
- #define [EBADE](#) 52
Invalid exchange.
- #define [EBADR](#) 53
Invalid request descriptor.
- #define [EXFULL](#) 54
Exchange full.
- #define [ENOANO](#) 55

- No anode.*
- #define [EBADRQC](#) 56
- Invalid request code.*
- #define [EBADSLT](#) 57
- Invalid slot.*
- #define [EDEADLOCK](#) [EDEADLK](#)
- Resource deadlock would occur.*
- #define [EBFONT](#) 59
- Bad font file format.*
- #define [ENOSTR](#) 60
- Device not a stream.*
- #define [ENODATA](#) 61
- No data available.*
- #define [ETIME](#) 62
- Timer expired.*
- #define [ENOSR](#) 63
- Out of streams resources.*
- #define [ENONET](#) 64
- Machine is not on the network.*
- #define [ENOPKG](#) 65
- Package not installed.*
- #define [EREMOTE](#) 66
- Object is remote.*
- #define [ENOLINK](#) 67
- Link has been severed.*
- #define [EADV](#) 68
- Advertise error.*
- #define [ESRMNT](#) 69
- Srmount error.*
- #define [ECOMM](#) 70
- Communication error on send.*
- #define [EPROTO](#) 71
- Protocol error.*
- #define [EMULTIHOP](#) 72
- Multihop attempted.*
- #define [EDOTDOT](#) 73
- RFS specific error.*
- #define [EBADMSG](#) 74
- Not a data message.*
- #define [EOVERFLOW](#) 75
- Value too large for defined data type.*
- #define [ENOTUNIQ](#) 76
- Name not unique on network.*

- #define [EBADFD](#) 77
File descriptor in bad state.
- #define [EREMCHG](#) 78
Remote address changed.
- #define [ELIBACC](#) 79
Can not access a needed shared library.
- #define [ELIBBAD](#) 80
Accessing a corrupted shared library.
- #define [ELIBSCN](#) 81
.lib section in a.out corrupted
- #define [ELIBMAX](#) 82
Attempting to link in too many shared libraries.
- #define [ELIBEXEC](#) 83
Cannot exec a shared library directly.
- #define [ELSEQ](#) 84
Illegal byte sequence.
- #define [ERESTART](#) 85
Interrupted system call should be restarted.
- #define [ESTRPIPE](#) 86
Streams pipe error.
- #define [EUSERS](#) 87
Too many users.
- #define [ENOTSOCK](#) 88
Socket operation on non-socket.
- #define [EDESTADDRREQ](#) 89
Destination address required.
- #define [EMSGSIZE](#) 90
Message too long.
- #define [EPROTOTYPE](#) 91
Protocol wrong type for socket.
- #define [ENOPROTOOPT](#) 92
Protocol not available.
- #define [EPROTONOSUPPORT](#) 93
Protocol not supported.
- #define [ESOCKTNOSUPPORT](#) 94
Socket type not supported.
- #define [EOPNOTSUPP](#) 95
Operation not supported on transport endpoint.
- #define [EPFNOSUPPORT](#) 96
Protocol family not supported.
- #define [EAFNOSUPPORT](#) 97
Address family not supported by protocol.
- #define [EADDRINUSE](#) 98

- Address already in use.*
- #define [EADDRNOTAVAIL](#) 99
- Cannot assign requested address.*
- #define [ENETDOWN](#) 100
- Network is down.*
- #define [ENETUNREACH](#) 101
- Network is unreachable.*
- #define [ENETRESET](#) 102
- Network dropped connection because of reset.*
- #define [ECONNABORTED](#) 103
- Software caused connection abort.*
- #define [ECONNRESET](#) 104
- Connection reset by peer.*
- #define [ENOBUFS](#) 105
- No buffer space available.*
- #define [EISCONN](#) 106
- Transport endpoint is already connected.*
- #define [ENOTCONN](#) 107
- Transport endpoint is not connected.*
- #define [ESHUTDOWN](#) 108
- Cannot send after transport endpoint shutdown.*
- #define [ETOOMANYREFS](#) 109
- Too many references: cannot splice.*
- #define [ETIMEDOUT](#) 110
- Connection timed out.*
- #define [ECONNREFUSED](#) 111
- Connection refused.*
- #define [EHOSTDOWN](#) 112
- Host is down.*
- #define [EHOSTUNREACH](#) 113
- No route to host.*
- #define [EALREADY](#) 114
- Operation already in progress.*
- #define [EINPROGRESS](#) 115
- Operation now in progress.*
- #define [ESTALE](#) 116
- Stale NFS file handle.*
- #define [EUCLEAN](#) 117
- Structure needs cleaning.*
- #define [ENOTNAM](#) 118
- Not a XENIX named type file.*
- #define [ENAVAIL](#) 119
- No XENIX semaphores available.*

- #define [EISNAM](#) 120
Is a named type file.
- #define [EREMOTEIO](#) 121
Remote I/O error.
- #define [EDQUOT](#) 122
Quota exceeded.
- #define [ENOMEDIUM](#) 123
No medium found.
- #define [EMEDIUMTYPE](#) 124
Wrong medium type.
- #define [ECANCELED](#) 125
Operation Canceled.
- #define [ENOKEY](#) 126
Required key not available.
- #define [EKEYEXPIRED](#) 127
Key has expired.
- #define [EKEYREVOKED](#) 128
Key has been revoked.
- #define [EKEYREJECTED](#) 129
Key was rejected by service.
- #define [EOWNERDEAD](#) 130
for robust mutexes
- #define [ENOTRECOVERABLE](#) 131
State not recoverable.
- #define [ERFKILL](#) 132
Operation not possible due to RF-kill.
- #define [EHWPOISON](#) 133
Memory page has hardware error.

4.1.1 Define Documentation

4.1.1.1 #define [E2BIG](#) 7

Argument list too long.

Definition at line 26 of file myerrno.h.

4.1.1.2 #define [EACCES](#) 13

Permission denied.

Definition at line 38 of file myerrno.h.

4.1.1.3 #define EADDRINUSE 98

Address already in use.

Definition at line 210 of file myerrno.h.

4.1.1.4 #define EADDRNOTAVAIL 99

Cannot assign requested address.

Definition at line 212 of file myerrno.h.

4.1.1.5 #define EADV 68

Advertise error.

Definition at line 150 of file myerrno.h.

4.1.1.6 #define EAFNOSUPPORT 97

Address family not supported by protocol.

Definition at line 208 of file myerrno.h.

4.1.1.7 #define EAGAIN 11

Try again.

Definition at line 34 of file myerrno.h.

4.1.1.8 #define EALREADY 114

Operation already in progress.

Definition at line 242 of file myerrno.h.

4.1.1.9 #define EBADE 52

Invalid exchange.

Definition at line 117 of file myerrno.h.

4.1.1.10 #define EBADF 9

Bad file number.

Definition at line 30 of file myerrno.h.

4.1.1.11 #define EBADFD 77

File descriptor in bad state.

Definition at line 168 of file myerrno.h.

4.1.1.12 #define EBADMSG 74

Not a data message.

Definition at line 162 of file myerrno.h.

4.1.1.13 #define EBADR 53

Invalid request descriptor.

Definition at line 119 of file myerrno.h.

4.1.1.14 #define EBADRQC 56

Invalid request code.

Definition at line 125 of file myerrno.h.

4.1.1.15 #define EBADSLT 57

Invalid slot.

Definition at line 127 of file myerrno.h.

4.1.1.16 #define EBFONT 59

Bad font file format.

Definition at line 132 of file myerrno.h.

4.1.1.17 #define EBUSY 16

Device or resource busy.

Definition at line 44 of file myerrno.h.

4.1.1.18 #define ECANCELED 125

Operation Canceled.

Definition at line 265 of file myerrno.h.

4.1.1.19 #define ECHILD 10

No child processes.

Definition at line 32 of file myerrno.h.

4.1.1.20 #define ECHRNG 44

Channel number out of range.

Definition at line 101 of file myerrno.h.

4.1.1.21 #define ECOMM 70

Communication error on send.

Definition at line 154 of file myerrno.h.

4.1.1.22 #define ECONNABORTED 103

Software caused connection abort.

Definition at line 220 of file myerrno.h.

4.1.1.23 #define ECONNREFUSED 111

Connection refused.

Definition at line 236 of file myerrno.h.

4.1.1.24 #define ECONNRESET 104

Connection reset by peer.

Definition at line 222 of file myerrno.h.

4.1.1.25 #define EDEADLK 35

Resource deadlock would occur.

Definition at line 83 of file myerrno.h.

4.1.1.26 #define EDEADLOCK EDEADLK

Resource deadlock would occur.

Definition at line 129 of file myerrno.h.

4.1.1.27 #define EDESTADDRREQ 89

Destination address required.

Definition at line 192 of file myerrno.h.

4.1.1.28 #define EDOM 33

Math argument out of domain of func.

Definition at line 78 of file myerrno.h.

4.1.1.29 #define EDOTDOT 73

RFS specific error.

Definition at line 160 of file myerrno.h.

4.1.1.30 #define EDQUOT 122

Quota exceeded.

Definition at line 258 of file myerrno.h.

4.1.1.31 #define EEXIST 17

File exists.

Definition at line 46 of file myerrno.h.

4.1.1.32 #define EFAULT 14

Bad address.

Definition at line 40 of file myerrno.h.

4.1.1.33 #define EFBIG 27

File too large.

Definition at line 66 of file myerrno.h.

4.1.1.34 #define EHOSTDOWN 112

Host is down.

Definition at line 238 of file myerrno.h.

4.1.1.35 #define EHOSTUNREACH 113

No route to host.

Definition at line 240 of file myerrno.h.

4.1.1.36 #define EHWPOISON 133

Memory page has hardware error.

Definition at line 286 of file myerrno.h.

4.1.1.37 #define EIDRM 43

Identifier removed.

Definition at line 99 of file myerrno.h.

4.1.1.38 #define EILSEQ 84

Illegal byte sequence.

Definition at line 182 of file myerrno.h.

4.1.1.39 #define EINPROGRESS 115

Operation now in progress.

Definition at line 244 of file myerrno.h.

4.1.1.40 #define EINTR 4

Interrupted system call.

Definition at line 20 of file myerrno.h.

4.1.1.41 #define EINVAL 22

Invalid argument.

Definition at line 56 of file myerrno.h.

4.1.1.42 #define EIO 5

I/O error.

Definition at line 22 of file myerrno.h.

4.1.1.43 #define EISCONN 106

Transport endpoint is already connected.

Definition at line 226 of file myerrno.h.

4.1.1.44 #define EISDIR 21

Is a directory.

Definition at line 54 of file myerrno.h.

4.1.1.45 #define EISNAM 120

Is a named type file.

Definition at line 254 of file myerrno.h.

4.1.1.46 #define EKEYEXPIRED 127

Key has expired.

Definition at line 269 of file myerrno.h.

4.1.1.47 #define EKEYREJECTED 129

Key was rejected by service.

Definition at line 273 of file myerrno.h.

4.1.1.48 #define EKEYREVOKED 128

Key has been revoked.

Definition at line 271 of file myerrno.h.

4.1.1.49 #define EL2HLT 51

Level 2 halted.

Definition at line 115 of file myerrno.h.

4.1.1.50 #define EL2NSYNC 45

Level 2 not synchronized.

Definition at line 103 of file myerrno.h.

4.1.1.51 #define EL3HLT 46

Level 3 halted.

Definition at line 105 of file myerrno.h.

4.1.1.52 #define EL3RST 47

Level 3 reset.

Definition at line 107 of file myerrno.h.

4.1.1.53 #define ELIBACC 79

Can not access a needed shared library.

Definition at line 172 of file myerrno.h.

4.1.1.54 #define ELIBBAD 80

Accessing a corrupted shared library.

Definition at line 174 of file myerrno.h.

4.1.1.55 #define ELIBEXEC 83

Cannot exec a shared library directly.

Definition at line 180 of file myerrno.h.

4.1.1.56 #define ELIBMAX 82

Attempting to link in too many shared libraries.

Definition at line 178 of file myerrno.h.

4.1.1.57 #define ELIBSCN 81

.lib section in a.out corrupted

Definition at line 176 of file myerrno.h.

4.1.1.58 #define ELNRNG 48

Link number out of range.

Definition at line 109 of file myerrno.h.

4.1.1.59 #define ELOOP 40

Too many symbolic links encountered.
Definition at line 93 of file myerrno.h.

4.1.1.60 #define EMEDIUMTYPE 124

Wrong medium type.
Definition at line 263 of file myerrno.h.

4.1.1.61 #define EMFILE 24

Too many open files.
Definition at line 60 of file myerrno.h.

4.1.1.62 #define EMLINK 31

Too many links.
Definition at line 74 of file myerrno.h.

4.1.1.63 #define EMSGSIZE 90

Message too long.
Definition at line 194 of file myerrno.h.

4.1.1.64 #define EMULTIHOP 72

Multihop attempted.
Definition at line 158 of file myerrno.h.

4.1.1.65 #define ENAMETOOLONG 36

File name too long.
Definition at line 85 of file myerrno.h.

4.1.1.66 #define ENAVAIL 119

No XENIX semaphores available.
Definition at line 252 of file myerrno.h.

4.1.1.67 #define ENETDOWN 100

Network is down.

Definition at line 214 of file myerrno.h.

4.1.1.68 #define ENETRESET 102

Network dropped connection because of reset.

Definition at line 218 of file myerrno.h.

4.1.1.69 #define ENETUNREACH 101

Network is unreachable.

Definition at line 216 of file myerrno.h.

4.1.1.70 #define ENFILE 23

File table overflow.

Definition at line 58 of file myerrno.h.

4.1.1.71 #define ENOANO 55

No anode.

Definition at line 123 of file myerrno.h.

4.1.1.72 #define ENOBUFS 105

No buffer space available.

Definition at line 224 of file myerrno.h.

4.1.1.73 #define ENOCSI 50

No CSI structure available.

Definition at line 113 of file myerrno.h.

4.1.1.74 #define ENODATA 61

No data available.

Definition at line 136 of file myerrno.h.

4.1.1.75 #define ENODEV 19

No such device.

Definition at line 50 of file myerrno.h.

4.1.1.76 #define ENOENT 2

No such file or directory.

Definition at line 16 of file myerrno.h.

4.1.1.77 #define ENOEXEC 8

Exec format error.

Definition at line 28 of file myerrno.h.

4.1.1.78 #define ENOKEY 126

Required key not available.

Definition at line 267 of file myerrno.h.

4.1.1.79 #define ENOLCK 37

No record locks available.

Definition at line 87 of file myerrno.h.

4.1.1.80 #define ENOLINK 67

Link has been severed.

Definition at line 148 of file myerrno.h.

4.1.1.81 #define ENOMEDIUM 123

No medium found.

Definition at line 261 of file myerrno.h.

4.1.1.82 #define ENOMEM 12

Out of memory.

Definition at line 36 of file myerrno.h.

4.1.1.83 #define ENOMSG 42

No message of desired type.

Definition at line 97 of file myerrno.h.

4.1.1.84 #define ENONET 64

Machine is not on the network.

Definition at line 142 of file myerrno.h.

4.1.1.85 #define ENOPKG 65

Package not installed.

Definition at line 144 of file myerrno.h.

4.1.1.86 #define ENOPROTOOPT 92

Protocol not available.

Definition at line 198 of file myerrno.h.

4.1.1.87 #define ENOSPC 28

No space left on device.

Definition at line 68 of file myerrno.h.

4.1.1.88 #define ENOSR 63

Out of streams resources.

Definition at line 140 of file myerrno.h.

4.1.1.89 #define ENOSTR 60

Device not a stream.

Definition at line 134 of file myerrno.h.

4.1.1.90 #define ENOSYS 38

Function not implemented.

Definition at line 89 of file myerrno.h.

4.1.1.91 #define ENOTBLK 15

Block device required.

Definition at line 42 of file myerrno.h.

4.1.1.92 #define ENOTCONN 107

Transport endpoint is not connected.

Definition at line 228 of file myerrno.h.

4.1.1.93 #define ENOTDIR 20

Not a directory.

Definition at line 52 of file myerrno.h.

4.1.1.94 #define ENOTEMPTY 39

Directory not empty.

Definition at line 91 of file myerrno.h.

4.1.1.95 #define ENOTNAM 118

Not a XENIX named type file.

Definition at line 250 of file myerrno.h.

4.1.1.96 #define ENOTRECOVERABLE 131

State not recoverable.

Definition at line 280 of file myerrno.h.

4.1.1.97 #define ENOTSOCK 88

Socket operation on non-socket.

Definition at line 190 of file myerrno.h.

4.1.1.98 #define ENOTTY 25

Not a typewriter.

Definition at line 62 of file myerrno.h.

4.1.1.99 #define ENOTUNIQ 76

Name not unique on network.

Definition at line 166 of file myerrno.h.

4.1.1.100 #define ENXIO 6

No such device or address.

Definition at line 24 of file myerrno.h.

4.1.1.101 #define EOPNOTSUPP 95

Operation not supported on transport endpoint.

Definition at line 204 of file myerrno.h.

4.1.1.102 #define EOVERFLOW 75

Value too large for defined data type.

Definition at line 164 of file myerrno.h.

4.1.1.103 #define EOWNERDEAD 130

for robust mutexes

Owner died

Definition at line 278 of file myerrno.h.

4.1.1.104 #define EPERM 1

Operation not permitted.

Definition at line 14 of file myerrno.h.

4.1.1.105 #define EPNOSUPPORT 96

Protocol family not supported.

Definition at line 206 of file myerrno.h.

4.1.1.106 #define EPIPE 32

Broken pipe.

Definition at line 76 of file myerrno.h.

4.1.1.107 #define EPROTO 71

Protocol error.

Definition at line 156 of file myerrno.h.

4.1.1.108 #define EPROTONOSUPPORT 93

Protocol not supported.

Definition at line 200 of file myerrno.h.

4.1.1.109 #define EPROTOTYPE 91

Protocol wrong type for socket.

Definition at line 196 of file myerrno.h.

4.1.1.110 #define ERANGE 34

Math result not representable.

Definition at line 80 of file myerrno.h.

4.1.1.111 #define EREMCHG 78

Remote address changed.

Definition at line 170 of file myerrno.h.

4.1.1.112 #define EREMOTE 66

Object is remote.

Definition at line 146 of file myerrno.h.

4.1.1.113 #define EREMOTEIO 121

Remote I/O error.

Definition at line 256 of file myerrno.h.

4.1.1.114 #define ERESTART 85

Interrupted system call should be restarted.

Definition at line 184 of file myerrno.h.

4.1.1.115 #define ERFKILL 132

Operation not possible due to RF-kill.

Definition at line 283 of file myerrno.h.

4.1.1.116 #define EROFS 30

Read-only file system.

Definition at line 72 of file myerrno.h.

4.1.1.117 #define ESHUTDOWN 108

Cannot send after transport endpoint shutdown.

Definition at line 230 of file myerrno.h.

4.1.1.118 #define ESOCKTNOSUPPORT 94

Socket type not supported.

Definition at line 202 of file myerrno.h.

4.1.1.119 #define EPIPE 29

Illegal seek.

Definition at line 70 of file myerrno.h.

4.1.1.120 #define ESRCH 3

No such process.

Definition at line 18 of file myerrno.h.

4.1.1.121 #define ESRMNT 69

Srmount error.

Definition at line 152 of file myerrno.h.

4.1.1.122 #define ESTALE 116

Stale NFS file handle.

Definition at line 246 of file myerrno.h.

4.1.1.123 #define ESTRPIPE 86

Streams pipe error.

Definition at line 186 of file myerrno.h.

4.1.1.124 #define ETIME 62

Timer expired.

Definition at line 138 of file myerrno.h.

4.1.1.125 #define ETIMEDOUT 110

Connection timed out.

Definition at line 234 of file myerrno.h.

4.1.1.126 #define ETOOMANYREFS 109

Too many references: cannot splice.

Definition at line 232 of file myerrno.h.

4.1.1.127 #define ETXTBSY 26

Text file busy.

Definition at line 64 of file myerrno.h.

4.1.1.128 #define EUCLEAN 117

Structure needs cleaning.

Definition at line 248 of file myerrno.h.

4.1.1.129 #define EUNATCH 49

Protocol driver not attached.

Definition at line 111 of file myerrno.h.

4.1.1.130 #define EUSERS 87

Too many users.

Definition at line 188 of file myerrno.h.

4.1.1.131 #define EWOLDBLOCK EAGAIN

Operation would block.

Definition at line 95 of file myerrno.h.

4.1.1.132 #define EXDEV 18

Cross-device link.

Definition at line 48 of file myerrno.h.

4.1.1.133 #define EXFULL 54

Exchange full.

Definition at line 121 of file myerrno.h.

4.1.1.134 #define SUCCESS 0

No Error.

Definition at line 11 of file myerrno.h.

Chapter 5

Data Structure Documentation

5.1 DIR Struct Reference

```
#include <pff.h>
```

Data Fields

- [WORD index](#)
- [BYTE * fn](#)
- [CLUST sclust](#)
- [CLUST clust](#)
- [DWORD sect](#)

5.1.1 Detailed Description

Definition at line 97 of file pff.h.

5.1.2 Field Documentation

5.1.2.1 CLUST DIR::clust

Definition at line 101 of file pff.h.

5.1.2.2 BYTE* DIR::fn

Definition at line 99 of file pff.h.

5.1.2.3 WORD DIR::index

Definition at line 98 of file pff.h.

5.1.2.4 CLUST DIR::sclust

Definition at line 100 of file pff.h.

5.1.2.5 DWORD DIR::sect

Definition at line 102 of file pff.h.

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

- [mmc/pff.h](#)

5.2 FATFS Struct Reference

```
#include <pff.h>
```

Data Fields

- [BYTE fs_type](#)
- [BYTE flag](#)
- [BYTE csize](#)
- [BYTE pad1](#)
- [WORD n_rootdir](#)
- [CLUST n_fatent](#)
- [DWORD fatbase](#)
- [DWORD dirbase](#)
- [DWORD database](#)
- [DWORD fptr](#)
- [DWORD fsize](#)
- [CLUST org_clust](#)
- [CLUST curr_clust](#)
- [DWORD dsect](#)

5.2.1 Detailed Description

Definition at line 76 of file pff.h.

5.2.2 Field Documentation

5.2.2.1 BYTE FATFS::csize

Definition at line 79 of file pff.h.

5.2.2.2 CLUST FATFS::curr_clust

Definition at line 89 of file pff.h.

5.2.2.3 DWORD FATFS::database

Definition at line 85 of file pff.h.

5.2.2.4 DWORD FATFS::dirbase

Definition at line 84 of file pff.h.

5.2.2.5 DWORD FATFS::dsect

Definition at line 90 of file pff.h.

5.2.2.6 DWORD FATFS::fatbase

Definition at line 83 of file pff.h.

5.2.2.7 BYTE FATFS::flag

Definition at line 78 of file pff.h.

5.2.2.8 DWORD FATFS::fptr

Definition at line 86 of file pff.h.

5.2.2.9 BYTE FATFS::fs_type

Definition at line 77 of file pff.h.

5.2.2.10 DWORD FATFS::fsize

Definition at line 87 of file pff.h.

5.2.2.11 CLUST FATFS::n_fatent

Definition at line 82 of file pff.h.

5.2.2.12 WORD FATFS::n_rootdir

Definition at line 81 of file pff.h.

5.2.2.13 CLUST FATFS::org_clust

Definition at line 88 of file pff.h.

5.2.2.14 BYTE FATFS::pad1

Definition at line 80 of file pff.h.

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

- [mmc/pff.h](#)

5.3 FILINFO Struct Reference

```
#include <pff.h>
```

Data Fields

- [DWORD fsize](#)
- [WORD fdate](#)
- [WORD ftime](#)
- [BYTE fattrib](#)
- [char fname](#) [13]

5.3.1 Detailed Description

Definition at line 109 of file pff.h.

5.3.2 Field Documentation

5.3.2.1 BYTE FILINFO::fattrib

Definition at line 113 of file pff.h.

5.3.2.2 WORD FILINFO::fdate

Definition at line 111 of file pff.h.

5.3.2.3 char FILINFO::fname[13]

Definition at line 114 of file pff.h.

5.3.2.4 DWORD FILINFO::fsize

Definition at line 110 of file pff.h.

5.3.2.5 WORD FILINFO::ftime

Definition at line 112 of file pff.h.

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

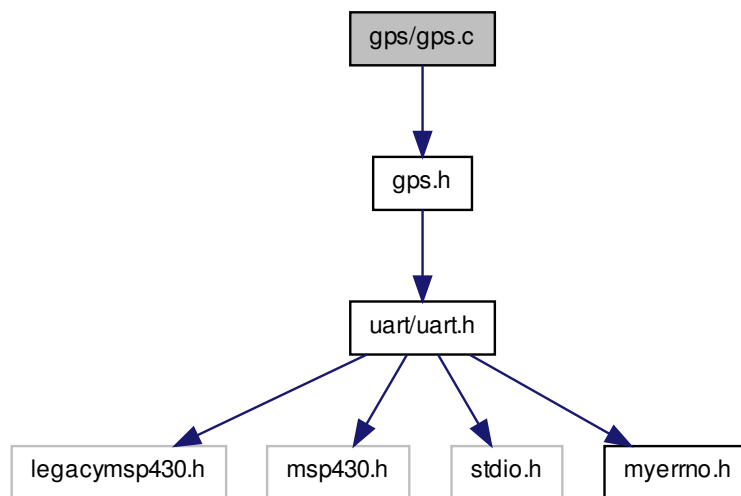
- mmc/[pff.h](#)

Chapter 6

File Documentation

6.1 gps/gps.c File Reference

`#include <gps.h>` Include dependency graph for `gps.c`:



Functions

- `int` [`gps_initialize`](#) (`void`)
Initialize the GPS and if necessary, the UART.

- `const char * gps_read (void)`

Read a data from the GPS.

6.1.1 Function Documentation

6.1.1.1 `int gps_initialize (void)`

Initialize the GPS and if necessary, the UART.

Definition at line 3 of file `gps.c`.

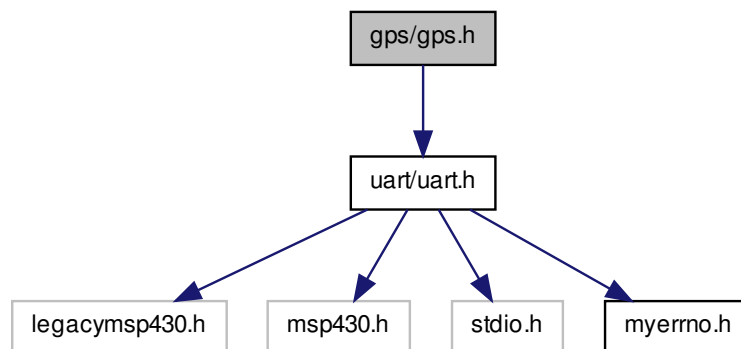
6.1.1.2 `const char* gps_read (void)`

Read a data from the GPS.

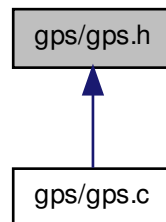
Definition at line 14 of file `gps.c`.

6.2 `gps/gps.h` File Reference

`#include <uart/uart.h>` Include dependency graph for `gps.h`:



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



Functions

- int [gps_initialize](#) (void)

Initialize the GPS and if necessary, the UART.

- const char * [gps_read](#) (void)

Read a data from the GPS.

6.2.1 Function Documentation

6.2.1.1 int [gps_initialize](#) (void)

Initialize the GPS and if necessary, the UART.

Definition at line 3 of file gps.c.

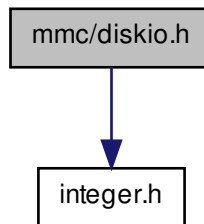
6.2.1.2 const char* [gps_read](#) (void)

Read a data from the GPS.

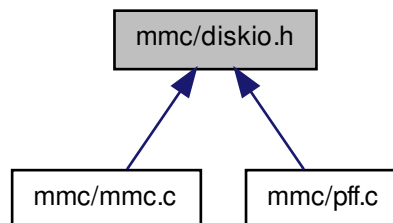
Definition at line 14 of file gps.c.

6.3 mmc/diskio.h File Reference

`#include "integer.h"` Include dependency graph for diskio.h:



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



Defines

- `#define STA_NOINIT` 0x01
- `#define STA_NODISK` 0x02
- `#define CT_MMC` 0x01
- `#define CT_SD1` 0x02
- `#define CT_SD2` 0x04
- `#define CT_SDC` (CT_SD1|CT_SD2)
- `#define CT_BLOCK` 0x08
- `#define _DISKIO`

Typedefs

- typedef [BYTE](#) [DSTATUS](#)

Enumerations

- enum [DRESULT](#) { [RES_OK](#) = 0, [RES_ERROR](#), [RES_NOTRDY](#), [RES_PARERR](#) }

Functions

- [DRESULT](#) [disk_initialize](#) (void)
- [DRESULT](#) [disk_readp](#) ([BYTE](#) *, [DWORD](#), [WORD](#), [WORD](#))
- [DRESULT](#) [disk_writep](#) (const [BYTE](#) *, [DWORD](#))

6.3.1 Define Documentation

6.3.1.1 #define [_DISKIO](#)

Definition at line 37 of file diskio.h.

6.3.1.2 #define [CT_BLOCK](#) 0x08

Block addressing

Definition at line 35 of file diskio.h.

6.3.1.3 #define [CT_MMC](#) 0x01

Card type flags (CardType) MMC ver 3

Definition at line 31 of file diskio.h.

6.3.1.4 #define [CT_SD1](#) 0x02

SD ver 1

Definition at line 32 of file diskio.h.

6.3.1.5 #define [CT_SD2](#) 0x04

SD ver 2

Definition at line 33 of file diskio.h.

6.3.1.6 `#define CT_SDC (CT_SD1|CT_SD2)`

SD

Definition at line 34 of file diskio.h.

6.3.1.7 `#define STA_NODISK 0x02`

No medium in the drive

Definition at line 28 of file diskio.h.

6.3.1.8 `#define STA_NOINIT 0x01`

Drive not initialized

Definition at line 27 of file diskio.h.

6.3.2 Typedef Documentation

6.3.2.1 `typedef BYTE DSTATUS`

Status of Disk Functions

Definition at line 10 of file diskio.h.

6.3.3 Enumeration Type Documentation

6.3.3.1 `enum DRESULT`

Results of Disk Functions

Enumerator:

RES_OK 0: Function succeeded

RES_ERROR 1: Disk error

RES_NOTRDY 2: Not ready

RES_PARERR 3: Invalid parameter

Definition at line 13 of file diskio.h.

6.3.4 Function Documentation

6.3.4.1 `DRESULT disk_initialize (void)`

Definition at line 88 of file mmc.c.

6.3.4.2 DRESULT disk_readp (BYTE *, DWORD , WORD , WORD)

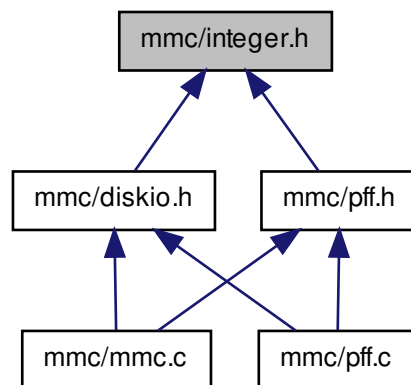
Definition at line 142 of file mmc.c.

6.3.4.3 DRESULT disk_writep (const BYTE *, DWORD)

Definition at line 203 of file mmc.c.

6.4 mmc/integer.h File Reference

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



Typedefs

- typedef int [INT](#)
- typedef unsigned int [UINT](#)
- typedef char [CHAR](#)
- typedef unsigned char [UCHAR](#)
- typedef unsigned char [BYTE](#)
- typedef short [SHORT](#)
- typedef unsigned short [USHORT](#)
- typedef unsigned short [WORD](#)
- typedef unsigned short [WCHAR](#)
- typedef long [LONG](#)
- typedef unsigned long [ULONG](#)
- typedef unsigned long [DWORD](#)

6.4.1 Typedef Documentation

6.4.1.1 typedef unsigned char **BYTE**

Definition at line 22 of file integer.h.

6.4.1.2 typedef char **CHAR**

These types must be 8-bit integer

Definition at line 20 of file integer.h.

6.4.1.3 typedef unsigned long **DWORD**

Definition at line 33 of file integer.h.

6.4.1.4 typedef int **INT**

< FatFs development platform These types must be 16-bit, 32-bit or larger integer

Definition at line 16 of file integer.h.

6.4.1.5 typedef long **LONG**

These types must be 32-bit integer

Definition at line 31 of file integer.h.

6.4.1.6 typedef short **SHORT**

These types must be 16-bit integer

Definition at line 25 of file integer.h.

6.4.1.7 typedef unsigned char **UCHAR**

Definition at line 21 of file integer.h.

6.4.1.8 typedef unsigned int **UINT**

Definition at line 17 of file integer.h.

6.4.1.9 typedef unsigned long **ULONG**

Definition at line 32 of file integer.h.

6.4.1.10 typedef unsigned short USHORT

Definition at line 26 of file integer.h.

6.4.1.11 typedef unsigned short WCHAR

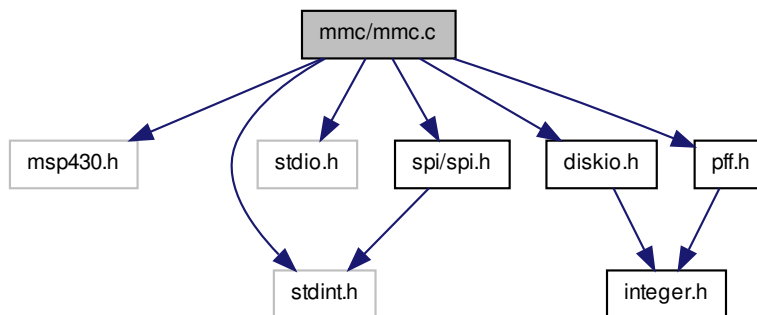
Definition at line 28 of file integer.h.

6.4.1.12 typedef unsigned short WORD

Definition at line 27 of file integer.h.

6.5 mmc/mmc.c File Reference

```
#include <msp430.h> #include <stdint.h> #include <stdio.h>
#include "diskio.h" #include "pff.h" #include "spi/spi.h"
#include "Include dependency graph for mmc.c:
```



Defines

- #define [DELAY_100US\(\)](#) __delay_cycles(1600)
- #define [SELECT\(\)](#) P2OUT &= ~BIT0
- #define [DESELECT\(\)](#) P2OUT |= BIT0
- #define [MMC_SEL](#) !(P2OUT & BIT0)
- #define [FORWARD\(d\)](#) [putchar\(d\)](#)
- #define [CMD0](#) (0x40+0)
- #define [CMD1](#) (0x40+1)
- #define [ACMD41](#) (0xC0+41)

- #define **CMD8** (0x40+8)
- #define **CMD16** (0x40+16)
- #define **CMD17** (0x40+17)
- #define **CMD24** (0x40+24)
- #define **CMD55** (0x40+55)
- #define **CMD58** (0x40+58)
- #define **CT_MMC** 0x01
- #define **CT_SD1** 0x02
- #define **CT_SD2** 0x04
- #define **CT_BLOCK** 0x08

Functions

- **DRESULT** `disk_initialize` (void)
- **DRESULT** `disk_readp` (**BYTE** *buff, **DWORD** lba, **WORD** ofs, **WORD** cnt)
- **DRESULT** `disk_writep` (const **BYTE** *buff, **DWORD** sa)

6.5.1 Define Documentation

6.5.1.1 #define **ACMD41** (0xC0+41)

SEND_OP_COND (SDC)

Definition at line 25 of file mmc.c.

6.5.1.2 #define **CMD0** (0x40+0)

GO_IDLE_STATE

Definition at line 23 of file mmc.c.

6.5.1.3 #define **CMD1** (0x40+1)

SEND_OP_COND (MMC)

Definition at line 24 of file mmc.c.

6.5.1.4 #define **CMD16** (0x40+16)

SET_BLOCKLEN

Definition at line 27 of file mmc.c.

6.5.1.5 #define **CMD17** (0x40+17)

READ_SINGLE_BLOCK

Definition at line 28 of file mmc.c.

6.5.1.6 #define CMD24 (0x40+24)

WRITE_BLOCK

Definition at line 29 of file mmc.c.

6.5.1.7 #define CMD55 (0x40+55)

APP_CMD

Definition at line 30 of file mmc.c.

6.5.1.8 #define CMD58 (0x40+58)

READ_OCR

Definition at line 31 of file mmc.c.

6.5.1.9 #define CMD8 (0x40+8)

SEND_IF_COND

Definition at line 26 of file mmc.c.

6.5.1.10 #define CT_BLOCK 0x08

Block addressing

Definition at line 37 of file mmc.c.

6.5.1.11 #define CT_MMC 0x01

MMC ver 3

Definition at line 34 of file mmc.c.

6.5.1.12 #define CT_SD1 0x02

SD ver 1

Definition at line 35 of file mmc.c.

6.5.1.13 #define CT_SD2 0x04

SD ver 2

Definition at line 36 of file mmc.c.

6.5.1.14 **#define DELAY_100US() __delay_cycles(1600)**

 ----- (100us/(1/16Mhz)) = 1600 ticks

Definition at line 16 of file mmc.c.

6.5.1.15 **#define DESELECT() P2OUT |= BIT0**

CS = H

Definition at line 18 of file mmc.c.

6.5.1.16 **#define FORWARD(d) putchar(d)**

Data forwarding function (Console out in this example)

Definition at line 20 of file mmc.c.

6.5.1.17 **#define MMC_SEL !(P2OUT & BIT0)**

CS status (true:CS == L)

Definition at line 19 of file mmc.c.

6.5.1.18 **#define SELECT() P2OUT &= ~BIT0**

CS = L

Definition at line 17 of file mmc.c.

6.5.2 Function Documentation

6.5.2.1 **DRESULT disk_initialize (void)**

 Definition at line 88 of file mmc.c.

6.5.2.2 **DRESULT disk_readp (BYTE * buff, DWORD lba, WORD ofs, WORD cnt)**

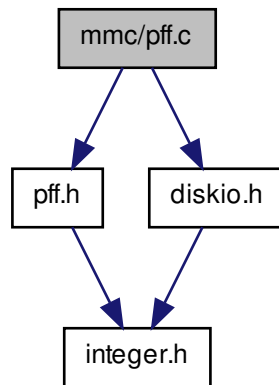
Definition at line 142 of file mmc.c.

6.5.2.3 **DRESULT disk_writep (const BYTE * buff, DWORD sa)**

Definition at line 203 of file mmc.c.

6.6 mmc/pff.c File Reference

#include "pff.h" #include "diskio.h" Include dependency graph for pff.c:



Defines

- #define `LD_CLUST(dir)` (((`DWORD`)`LD_WORD`(`dir+DIR_FstClusHI`)<<16) | `LD_WORD`(`dir+DIR_FstClusLO`))
- #define `_DF1S` 0
- #define `IsUpper(c)` (((`c`)>='A')&&((`c`)<='Z'))
- #define `IsLower(c)` (((`c`)>='a')&&((`c`)<='z'))
- #define `IsDBCS1(c)` 0
- #define `IsDBCS2(c)` 0
- #define `BS_jumpBoot` 0
- #define `BS_OEMName` 3
- #define `BPB_BytsPerSec` 11
- #define `BPB_SecPerClus` 13
- #define `BPB_RsvdSecCnt` 14
- #define `BPB_NumFATs` 16
- #define `BPB_RootEntCnt` 17
- #define `BPB_TotSec16` 19
- #define `BPB_Media` 21
- #define `BPB_FATs16` 22
- #define `BPB_SecPerTrk` 24
- #define `BPB_NumHeads` 26

- `#define BPB_HiddSec` 28
- `#define BPB_TotSec32` 32
- `#define BS_55AA` 510
- `#define BS_DrvNum` 36
- `#define BS_BootSig` 38
- `#define BS_VolID` 39
- `#define BS_VolLab` 43
- `#define BS_FilSysType` 54
- `#define BPB_FATSz32` 36
- `#define BPB_ExtFlags` 40
- `#define BPB_FSVer` 42
- `#define BPB_RootClus` 44
- `#define BPB_FSInfo` 48
- `#define BPB_BkBootSec` 50
- `#define BS_DrvNum32` 64
- `#define BS_BootSig32` 66
- `#define BS_VolID32` 67
- `#define BS_VolLab32` 71
- `#define BS_FilSysType32` 82
- `#define MBR_Table` 446
- `#define DIR_Name` 0
- `#define DIR_Attr` 11
- `#define DIR_NTres` 12
- `#define DIR_CrtTime` 14
- `#define DIR_CrtDate` 16
- `#define DIR_FstClusHI` 20
- `#define DIR_WrtTime` 22
- `#define DIR_WrtDate` 24
- `#define DIR_FstClusLO` 26
- `#define DIR_FileSize` 28

Functions

- `FRESULT pf_mount` (FATFS *fs)
- `FRESULT pf_open` (const char *path)
- `FRESULT pf_read` (void *buff, WORD btr, WORD *br)
- `FRESULT pf_write` (const void *buff, WORD btw, WORD *bw)
- `FRESULT pf_lseek` (DWORD ofs)
- `FRESULT pf_opendir` (DIR *dj, const char *path)
- `FRESULT pf_readdir` (DIR *dj, FILINFO *fno)

6.6.1 Define Documentation

6.6.1.1 `#define _DF1S` 0

Definition at line 231 of file pff.c.

6.6.1.2 #define BPB_BkBootSec 50

Definition at line 298 of file pff.c.

6.6.1.3 #define BPB_BytsPerSec 11

Definition at line 273 of file pff.c.

6.6.1.4 #define BPB_ExtFlags 40

Definition at line 294 of file pff.c.

6.6.1.5 #define BPB_FATSz16 22

Definition at line 280 of file pff.c.

6.6.1.6 #define BPB_FATSz32 36

Definition at line 293 of file pff.c.

6.6.1.7 #define BPB_FSInfo 48

Definition at line 297 of file pff.c.

6.6.1.8 #define BPB_FSVer 42

Definition at line 295 of file pff.c.

6.6.1.9 #define BPB_HiddSec 28

Definition at line 283 of file pff.c.

6.6.1.10 #define BPB_Media 21

Definition at line 279 of file pff.c.

6.6.1.11 #define BPB_NumFATs 16

Definition at line 276 of file pff.c.

6.6.1.12 #define BPB_NumHeads 26

Definition at line 282 of file pff.c.

6.6.1.13 #define BPB_RootClus 44

Definition at line 296 of file pff.c.

6.6.1.14 #define BPB_RootEntCnt 17

Definition at line 277 of file pff.c.

6.6.1.15 #define BPB_RsvdSecCnt 14

Definition at line 275 of file pff.c.

6.6.1.16 #define BPB_SecPerClus 13

Definition at line 274 of file pff.c.

6.6.1.17 #define BPB_SecPerTrk 24

Definition at line 281 of file pff.c.

6.6.1.18 #define BPB_TotSec16 19

Definition at line 278 of file pff.c.

6.6.1.19 #define BPB_TotSec32 32

Definition at line 284 of file pff.c.

6.6.1.20 #define BS_55AA 510

Definition at line 285 of file pff.c.

6.6.1.21 #define BS_BootSig 38

Definition at line 288 of file pff.c.

6.6.1.22 #define BS_BootSig32 66

Definition at line 300 of file pff.c.

6.6.1.23 #define BS_DrvNum 36

Definition at line 287 of file pff.c.

6.6.1.24 #define BS_DrvNum32 64

Definition at line 299 of file pff.c.

6.6.1.25 #define BS_FilSysType 54

Definition at line 291 of file pff.c.

6.6.1.26 #define BS_FilSysType32 82

Definition at line 303 of file pff.c.

6.6.1.27 #define BS_jmpBoot 0

Definition at line 271 of file pff.c.

6.6.1.28 #define BS_OEMName 3

Definition at line 272 of file pff.c.

6.6.1.29 #define BS_VolID 39

Definition at line 289 of file pff.c.

6.6.1.30 #define BS_VolID32 67

Definition at line 301 of file pff.c.

6.6.1.31 #define BS_VolLab 43

Definition at line 290 of file pff.c.

6.6.1.32 **#define BS_VolLab32** 71

Definition at line 302 of file pff.c.

6.6.1.33 **#define DIR_Attr** 11

Definition at line 308 of file pff.c.

6.6.1.34 **#define DIR_CrtDate** 16

Definition at line 311 of file pff.c.

6.6.1.35 **#define DIR_CrtTime** 14

Definition at line 310 of file pff.c.

6.6.1.36 **#define DIR_FileSize** 28

Definition at line 316 of file pff.c.

6.6.1.37 **#define DIR_FstClusHI** 20

Definition at line 312 of file pff.c.

6.6.1.38 **#define DIR_FstClusLO** 26

Definition at line 315 of file pff.c.

6.6.1.39 **#define DIR_Name** 0

Definition at line 307 of file pff.c.

6.6.1.40 **#define DIR_NTres** 12

Definition at line 309 of file pff.c.

6.6.1.41 **#define DIR_WrtDate** 24

Definition at line 314 of file pff.c.

6.6.1.42 **#define DIR_WrtTime** 22

Definition at line 313 of file pff.c.

6.6.1.43 **#define IsDBCS1(c)** 0

Definition at line 261 of file pff.c.

6.6.1.44 **#define IsDBCS2(c)** 0

Definition at line 262 of file pff.c.

6.6.1.45 **#define IsLower(c)** (((c)>='a')&&((c)<='z'))

Definition at line 243 of file pff.c.

6.6.1.46 **#define IsUpper(c)** (((c)>='A')&&((c)<='Z'))

Definition at line 242 of file pff.c.

6.6.1.47 **#define LD_CLUST(dir)** (((DWORD)LD_WORD(dir+DIR_FstClusHI)<<16) |
LD_WORD(dir+DIR_FstClusLO))

Definition at line 38 of file pff.c.

6.6.1.48 **#define MBR_Table** 446

Definition at line 305 of file pff.c.

6.6.2 Function Documentation

6.6.2.1 **FRESULT pf_lseek (DWORD ofs)**

Definition at line 985 of file pff.c.

6.6.2.2 **FRESULT pf_mount (FATFS * fs)**

Definition at line 738 of file pff.c.

6.6.2.3 **FRESULT pf_open (const char * path)**

Definition at line 820 of file pff.c.

6.6.2.4 FRESULT pf_opendir (DIR * *dj*, const char * *path*)

Definition at line 1040 of file pff.c.

6.6.2.5 FRESULT pf_read (void * *buff*, WORD *btr*, WORD * *br*)

Definition at line 856 of file pff.c.

6.6.2.6 FRESULT pf_readdir (DIR * *dj*, FILINFO * *fno*)

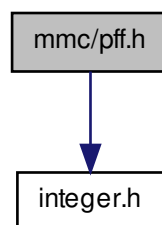
Definition at line 1078 of file pff.c.

6.6.2.7 FRESULT pf_write (const void * *buff*, WORD *btw*, WORD * *bw*)

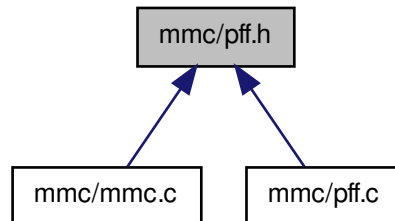
Definition at line 914 of file pff.c.

6.7 mmc/pff.h File Reference

#include "integer.h" Include dependency graph for pff.h:



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



Data Structures

- struct [FATFS](#)
- struct [DIR](#)
- struct [FILINFO](#)

Defines

- `#define _USE_READ 1 /* 1:Enable pf_read\(\) */`
- `#define _USE_DIR 1 /* 1:Enable pf_opendir\(\) and pf_readdir\(\) */`
- `#define _USE_LSEEK 1 /* 1:Enable pf_lseek\(\) */`
- `#define _USE_WRITE 1 /* 1:Enable pf_write\(\) */`
- `#define _FS_FAT12 0 /* 1:Enable FAT12 support */`
- `#define _FS_FAT32 1 /* 1:Enable FAT32 support */`
- `#define _CODE_PAGE 1`
- `#define _WORD_ACCESS 0`
- `#define CLUST_DWORD`
- `#define FA_OPENED 0x01`
- `#define FA_WPRT 0x02`
- `#define FA__WIP 0x40`
- `#define FS_FAT12 1`
- `#define FS_FAT16 2`
- `#define FS_FAT32 3`
- `#define AM_RDO 0x01 /* Read only */`
- `#define AM_HID 0x02 /* Hidden */`
- `#define AM_SYS 0x04 /* System */`
- `#define AM_VOL 0x08 /* Volume label */`
- `#define AM_LFN 0x0F /* LFN entry */`
- `#define AM_DIR 0x10 /* Directory */`

- `#define AM_ARC 0x20 /* Archive */`
- `#define AM_MASK 0x3F /* Mask of defined bits */`
- `#define LD_WORD(ptr) (WORD)(((WORD)*((BYTE*)(ptr)+1)<<8)|(WORD)*((BYTE*)(ptr))`
- `#define LD_DWORD(ptr) (DWORD)(((DWORD)*((BYTE*)(ptr)+3)<<24)|((-DWORD)*((BYTE*)(ptr)+2)<<16)|((WORD)*((BYTE*)(ptr)+1)<<8)|*(BYTE*)(ptr))`
- `#define ST_WORD(ptr, val) *((BYTE*)(ptr))=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8)`
- `#define ST_DWORD(ptr, val) *((BYTE*)(ptr))=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8); *((BYTE*)(ptr)+2)=(BYTE)((DWORD)(val)>>16); *((BYTE*)(ptr)+3)=(BYTE)((DWORD)(val)>>24)`

Enumerations

- `enum FRESULT { FR_OK = 0, FR_DISK_ERR, FR_NOT_READY, FR_NO_FILE, FR_NO_PATH, FR_NOT_OPENED, FR_NOT_ENABLED, FR_NO_FILESYSTEM }`

Functions

- `FRESULT pf_mount (FATFS *)`
- `FRESULT pf_open (const char *)`
- `FRESULT pf_read (void *, WORD, WORD *)`
- `FRESULT pf_write (const void *, WORD, WORD *)`
- `FRESULT pf_lseek (DWORD)`
- `FRESULT pf_opendir (DIR *, const char *)`
- `FRESULT pf_readdir (DIR *, FILINFO *)`

6.7.1 Define Documentation

6.7.1.1 `#define _CODE_PAGE 1`

Definition at line 41 of file pff.h.

6.7.1.2 `#define _FS_FAT12 0 /* 1:Enable FAT12 support */`

Definition at line 37 of file pff.h.

6.7.1.3 `#define _FS_FAT32 1 /* 1:Enable FAT32 support */`

Definition at line 38 of file pff.h.

6.7.1.4 **#define _USE_DIR 1** /* 1:Enable pf_opendir() and pf_readdir() */

Definition at line 31 of file pff.h.

6.7.1.5 **#define _USE_LSEEK 1** /* 1:Enable pf_lseek() */

Definition at line 33 of file pff.h.

6.7.1.6 **#define _USE_READ 1** /* 1:Enable pf_read() */

Definition at line 29 of file pff.h.

6.7.1.7 **#define _USE_WRITE 1** /* 1:Enable pf_write() */

Definition at line 35 of file pff.h.

6.7.1.8 **#define _WORD_ACCESS 0**

Definition at line 49 of file pff.h.

6.7.1.9 **#define AM_ARC 0x20** /* Archive */

Definition at line 172 of file pff.h.

6.7.1.10 **#define AM_DIR 0x10** /* Directory */

Definition at line 171 of file pff.h.

6.7.1.11 **#define AM_HID 0x02** /* Hidden */

Definition at line 167 of file pff.h.

6.7.1.12 **#define AM_LFN 0x0F** /* LFN entry */

Definition at line 170 of file pff.h.

6.7.1.13 **#define AM_MASK 0x3F** /* Mask of defined bits */

Definition at line 173 of file pff.h.

6.7.1.14 **#define AM_RDO 0x01 /* Read only */**

Definition at line 166 of file pff.h.

6.7.1.15 **#define AM_SYS 0x04 /* System */**

Definition at line 168 of file pff.h.

6.7.1.16 **#define AM_VOL 0x08 /* Volume label */**

Definition at line 169 of file pff.h.

6.7.1.17 **#define CLUST DWORD**

Definition at line 68 of file pff.h.

6.7.1.18 **#define FA_WIP 0x40**

Definition at line 154 of file pff.h.

6.7.1.19 **#define FA_OPENED 0x01**

Definition at line 152 of file pff.h.

6.7.1.20 **#define FA_WPRT 0x02**

Definition at line 153 of file pff.h.

6.7.1.21 **#define FS_FAT12 1**

Definition at line 159 of file pff.h.

6.7.1.22 **#define FS_FAT16 2**

Definition at line 160 of file pff.h.

6.7.1.23 **#define FS_FAT32 3**

Definition at line 161 of file pff.h.

6.7.1.24 **#define LD_DWORD(ptr)** (DWORD)((DWORD)*((BYTE*)(ptr)+3)<<24)|((DWORD)*((BYTE*)(ptr)+2)<<16)|((WORD)*((BYTE*)(ptr)+1)<<8)|*(BYTE*)(ptr))

Definition at line 186 of file pff.h.

6.7.1.25 **#define LD_WORD(ptr)** (WORD)((WORD)*((BYTE*)(ptr)+1)<<8)|((WORD)*((BYTE*)(ptr))

Definition at line 185 of file pff.h.

6.7.1.26 **#define ST_DWORD(ptr, val)** *(BYTE*)(ptr)=(BYTE)(val); *((BYTE*)(ptr)+1)=(-BYTE)((WORD)(val)>>8); *((BYTE*)(ptr)+2)=(BYTE)((DWORD)(val)>>16); *((BYTE*)(ptr)+3)=(BYTE)((DWORD)(val)>>24)

Definition at line 188 of file pff.h.

6.7.1.27 **#define ST_WORD(ptr, val)** *(BYTE*)(ptr)=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8)

Definition at line 187 of file pff.h.

6.7.2 Enumeration Type Documentation

6.7.2.1 enum FRESULT

Enumerator:

FR_OK
FR_DISK_ERR
FR_NOT_READY
FR_NO_FILE
FR_NO_PATH
FR_NOT_OPENED
FR_NOT_ENABLED
FR_NO_FILESYSTEM

Definition at line 121 of file pff.h.

6.7.3 Function Documentation

6.7.3.1 FRESULT pf_lseek (DWORD)

Definition at line 985 of file pff.c.

6.7.3.2 FRESULT pf_mount (FATFS *)

Definition at line 738 of file pff.c.

6.7.3.3 FRESULT pf_open (const char *)

Definition at line 820 of file pff.c.

6.7.3.4 FRESULT pf_opendir (DIR *, const char *)

Definition at line 1040 of file pff.c.

6.7.3.5 FRESULT pf_read (void *, WORD , WORD *)

Definition at line 856 of file pff.c.

6.7.3.6 FRESULT pf_readdir (DIR *, FILINFO *)

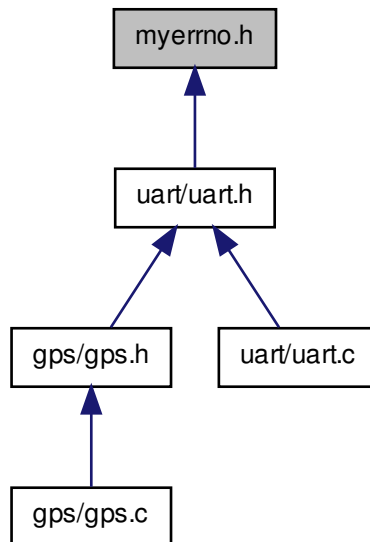
Definition at line 1078 of file pff.c.

6.7.3.7 FRESULT pf_write (const void *, WORD , WORD *)

Definition at line 914 of file pff.c.

6.8 myerrno.h File Reference

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



Defines

- #define **SUCCESS** 0
No Error.
- #define **EPERM** 1
Operation not permitted.
- #define **ENOENT** 2
No such file or directory.
- #define **ESRCH** 3
No such process.
- #define **EINTR** 4
Interrupted system call.
- #define **EIO** 5
I/O error.
- #define **ENXIO** 6
No such device or address.
- #define **E2BIG** 7

- Argument list too long.*
- #define [ENOEXEC](#) 8
- Exec format error.*
- #define [EBADF](#) 9
- Bad file number.*
- #define [ECHILD](#) 10
- No child processes.*
- #define [EAGAIN](#) 11
- Try again.*
- #define [ENOMEM](#) 12
- Out of memory.*
- #define [EACCES](#) 13
- Permission denied.*
- #define [EFAULT](#) 14
- Bad address.*
- #define [ENOTBLK](#) 15
- Block device required.*
- #define [EBUSY](#) 16
- Device or resource busy.*
- #define [EEXIST](#) 17
- File exists.*
- #define [EXDEV](#) 18
- Cross-device link.*
- #define [ENODEV](#) 19
- No such device.*
- #define [ENOTDIR](#) 20
- Not a directory.*
- #define [EISDIR](#) 21
- Is a directory.*
- #define [EINVAL](#) 22
- Invalid argument.*
- #define [ENFILE](#) 23
- File table overflow.*
- #define [EMFILE](#) 24
- Too many open files.*
- #define [ENOTTY](#) 25
- Not a typewriter.*
- #define [ETXTBSY](#) 26
- Text file busy.*
- #define [EBIG](#) 27
- File too large.*
- #define [ENOSPC](#) 28
- No space left on device.*

- #define [ESPIPE](#) 29
Illegal seek.
- #define [EROFS](#) 30
Read-only file system.
- #define [EMLINK](#) 31
Too many links.
- #define [EPIPE](#) 32
Broken pipe.
- #define [EDOM](#) 33
Math argument out of domain of func.
- #define [ERANGE](#) 34
Math result not representable.
- #define [EDEADLK](#) 35
Resource deadlock would occur.
- #define [ENAMETOOLONG](#) 36
File name too long.
- #define [ENOLCK](#) 37
No record locks available.
- #define [ENOSYS](#) 38
Function not implemented.
- #define [ENOTEMPTY](#) 39
Directory not empty.
- #define [ELOOP](#) 40
Too many symbolic links encountered.
- #define [EWOULDBLOCK](#) [EAGAIN](#)
Operation would block.
- #define [ENOMSG](#) 42
No message of desired type.
- #define [EIDRM](#) 43
Identifier removed.
- #define [ECHRNG](#) 44
Channel number out of range.
- #define [EL2NSYNC](#) 45
Level 2 not synchronized.
- #define [EL3HLT](#) 46
Level 3 halted.
- #define [EL3RST](#) 47
Level 3 reset.
- #define [ELNRNG](#) 48
Link number out of range.
- #define [EUNATCH](#) 49
Protocol driver not attached.
- #define [ENOCSS](#) 50

- No CSI structure available.*
- #define [EL2HLT](#) 51
 - Level 2 halted.*
- #define [EBADE](#) 52
 - Invalid exchange.*
- #define [EBADR](#) 53
 - Invalid request descriptor.*
- #define [EXFULL](#) 54
 - Exchange full.*
- #define [ENOANO](#) 55
 - No anode.*
- #define [EBADRQC](#) 56
 - Invalid request code.*
- #define [EBADSLT](#) 57
 - Invalid slot.*
- #define [EDEADLOCK](#) [EDEADLK](#)
 - Resource deadlock would occur.*
- #define [EBFONT](#) 59
 - Bad font file format.*
- #define [ENOSTR](#) 60
 - Device not a stream.*
- #define [ENODATA](#) 61
 - No data available.*
- #define [ETIME](#) 62
 - Timer expired.*
- #define [ENOSR](#) 63
 - Out of streams resources.*
- #define [ENONET](#) 64
 - Machine is not on the network.*
- #define [ENOPKG](#) 65
 - Package not installed.*
- #define [EREMOTE](#) 66
 - Object is remote.*
- #define [ENOLINK](#) 67
 - Link has been severed.*
- #define [EADV](#) 68
 - Advertise error.*
- #define [ESRMNT](#) 69
 - Srmount error.*
- #define [ECOMM](#) 70
 - Communication error on send.*
- #define [EPROTO](#) 71
 - Protocol error.*

- #define [EMULTIHOP](#) 72
Multihop attempted.
- #define [EDOTDOT](#) 73
RFS specific error.
- #define [EBADMSG](#) 74
Not a data message.
- #define [EOVERFLOW](#) 75
Value too large for defined data type.
- #define [ENOTUNIQ](#) 76
Name not unique on network.
- #define [EBADFD](#) 77
File descriptor in bad state.
- #define [EREMCHG](#) 78
Remote address changed.
- #define [ELIBACC](#) 79
Can not access a needed shared library.
- #define [ELIBBAD](#) 80
Accessing a corrupted shared library.
- #define [ELIBSCN](#) 81
.lib section in a.out corrupted
- #define [ELIBMAX](#) 82
Attempting to link in too many shared libraries.
- #define [ELIBEXEC](#) 83
Cannot exec a shared library directly.
- #define [EILSEQ](#) 84
Illegal byte sequence.
- #define [ERESTART](#) 85
Interrupted system call should be restarted.
- #define [ESTRPIPE](#) 86
Streams pipe error.
- #define [EUSERS](#) 87
Too many users.
- #define [ENOTSOCK](#) 88
Socket operation on non-socket.
- #define [EDESTADDRREQ](#) 89
Destination address required.
- #define [EMSGSIZE](#) 90
Message too long.
- #define [EPROTOTYPE](#) 91
Protocol wrong type for socket.
- #define [ENOPROTOOPT](#) 92
Protocol not available.
- #define [EPROTONOSUPPORT](#) 93

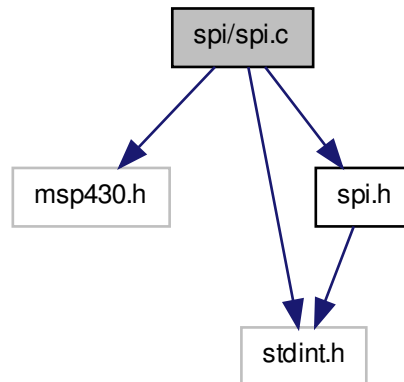
- Protocol not supported.*
- #define [ESOCKTNOSUPPORT](#) 94
- Socket type not supported.*
- #define [EOPNOTSUPP](#) 95
- Operation not supported on transport endpoint.*
- #define [EPFNOSUPPORT](#) 96
- Protocol family not supported.*
- #define [EAFNOSUPPORT](#) 97
- Address family not supported by protocol.*
- #define [EADDRINUSE](#) 98
- Address already in use.*
- #define [EADDRNOTAVAIL](#) 99
- Cannot assign requested address.*
- #define [ENETDOWN](#) 100
- Network is down.*
- #define [ENETUNREACH](#) 101
- Network is unreachable.*
- #define [ENETRESET](#) 102
- Network dropped connection because of reset.*
- #define [ECONNABORTED](#) 103
- Software caused connection abort.*
- #define [ECONNRESET](#) 104
- Connection reset by peer.*
- #define [ENOBUFS](#) 105
- No buffer space available.*
- #define [EISCONN](#) 106
- Transport endpoint is already connected.*
- #define [ENOTCONN](#) 107
- Transport endpoint is not connected.*
- #define [ESHUTDOWN](#) 108
- Cannot send after transport endpoint shutdown.*
- #define [ETOOMANYREFS](#) 109
- Too many references: cannot splice.*
- #define [ETIMEDOUT](#) 110
- Connection timed out.*
- #define [ECONNREFUSED](#) 111
- Connection refused.*
- #define [EHOSTDOWN](#) 112
- Host is down.*
- #define [EHOSTUNREACH](#) 113
- No route to host.*
- #define [EALREADY](#) 114
- Operation already in progress.*

- #define [EINPROGRESS](#) 115
Operation now in progress.
- #define [ESTALE](#) 116
Stale NFS file handle.
- #define [EUCLEAN](#) 117
Structure needs cleaning.
- #define [ENOTNAM](#) 118
Not a XENIX named type file.
- #define [ENAVAIL](#) 119
No XENIX semaphores available.
- #define [EISNAM](#) 120
Is a named type file.
- #define [EREMOTEIO](#) 121
Remote I/O error.
- #define [EDQUOT](#) 122
Quota exceeded.
- #define [ENOMEDIUM](#) 123
No medium found.
- #define [EMEDIUMTYPE](#) 124
Wrong medium type.
- #define [ECANCELED](#) 125
Operation Canceled.
- #define [ENOKEY](#) 126
Required key not available.
- #define [EKEYEXPIRED](#) 127
Key has expired.
- #define [EKEYREVOKED](#) 128
Key has been revoked.
- #define [EKEYREJECTED](#) 129
Key was rejected by service.
- #define [EOWNERDEAD](#) 130
for robust mutexes
- #define [ENOTRECOVERABLE](#) 131
State not recoverable.
- #define [ERFKILL](#) 132
Operation not possible due to RF-kill.
- #define [EHWPOISON](#) 133
Memory page has hardware error.

6.9 spi/spi.c File Reference

```
#include <msp430.h> #include <stdint.h> #include "spi.h" ×
```

Include dependency graph for spi.c:



Defines

- `#define SPI_MODE_0 (UCMSB | UCMST | UCSYNC | UCCKPH) /* CPOL=0 CPHA=0 */`
- `#define SPI_MODE_1 (UCMSB | UCMST | UCSYNC) /* CPOL=0 CPHA=1 */`
- `#define SPI_MODE_2 (UCMSB | UCMST | UCSYNC | UCCKPL | UCCKPH) /* CPOL=1 CPHA=0 */`
- `#define SPI_MODE_3 (UCMSB | UCMST | UCSYNC | UCCKPL) /* CPOL=1 CPHA=1 */`
- `#define LOBYTE(w) ((w)&0xFF)`
- `#define HIBYTE(w) ((w)>>8)`

Functions

- `void spi_initialize (void)`
- `uint8_t spi_send (const uint8_t c)`
- `uint8_t spi_receive (void)`
- `uint16_t spi_set_divisor (const uint16_t clkdiv)`

6.9.1 Define Documentation

6.9.1.1 `#define HIBYTE(w) ((w)>>8)`

Definition at line 31 of file spi.c.

6.9.1.2 `#define LOBYTE(w) ((w)&0xFF)`

utility macros for extracting hi/lo byte data from a word value

Definition at line 30 of file spi.c.

6.9.1.3 `#define SPI_MODE_0 (UCMSB | UCMST | UCSYNC | UCCKPH) /* CPOL=0 CPHA=0 */`

File: usci_spi.c - msp430 USCI SPI implementation USCI flags for various the SPI MODEs

Note: The msp430 UCCKPL tracks the CPOL value. However, the UCCKPH flag is inverted when compared to the CPHA value described in Motorola documentation.

Definition at line 21 of file spi.c.

6.9.1.4 `#define SPI_MODE_1 (UCMSB | UCMST | UCSYNC) /* CPOL=0 CPHA=1 */`

Definition at line 22 of file spi.c.

6.9.1.5 `#define SPI_MODE_2 (UCMSB | UCMST | UCSYNC | UCCKPL | UCCKPH) /* CPOL=1 CPHA=0 */`

Definition at line 23 of file spi.c.

6.9.1.6 `#define SPI_MODE_3 (UCMSB | UCMST | UCSYNC | UCCKPL) /* CPOL=1 CPHA=1 */`

Definition at line 24 of file spi.c.

6.9.2 Function Documentation

6.9.2.1 `void spi_initialize (void)`

[spi_initialize\(\)](#) - Configure USCI UCB0 for SPI mode

P2.0 - CS (active low) P1.5 - SCLK P1.6 - SIMO/MOSI P1.7 - SOMI/MISO

Definition at line 42 of file spi.c.

6.9.2.2 `uint8_t spi_receive (void)`

`spi_receive()` - send dummy byte then recv response

Definition at line 80 of file `spi.c`.

6.9.2.3 `uint8_t spi_send (const uint8_t c)`

`spi_send()` - send a byte and recv response

Definition at line 64 of file `spi.c`.

6.9.2.4 `uint16_t spi_set_divisor (const uint16_t clkdiv)`

`spi_set_divisor()` - set new clock divider for USCI

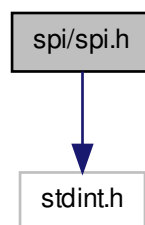
USCI speed is based on the SMCLK divided by BR0 and BR1 initially we start slow (400kHz) to conform to SDCard specifications then we speed up once initialized (SPI_DEFAULT_SPEED)

returns the previous setting

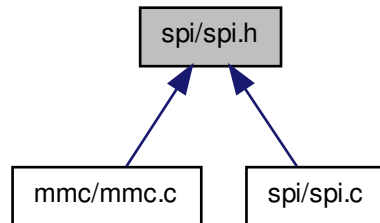
Definition at line 103 of file `spi.c`.

6.10 `spi/spi.h` File Reference

`#include <stdint.h>` Include dependency graph for `spi.h`:



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



Defines

- `#define SPI_250kHz 64 /* 16MHz/250000 */`
- `#define SPI_400kHz 40 /* 16MHz/400000 */`
- `#define SPI_1MHz 16 /* 16MHz/1MHz */`
- `#define SPI_2MHz 8 /* 16MHz/2MHz */`
- `#define SPI_4MHz 4 /* 16MHz/4MHz */`
- `#define SPI_8MHz 2 /* 16MHz/8MHz */`
- `#define SPI_16MHz 1 /* 16MHz/16Mhz */`
- `#define SPI_DEFAULT_SPEED SPI_8MHz`

Functions

- `void spi_initialize (void)`
- `uint8_t spi_send (const uint8_t)`
- `uint8_t spi_receive (void)`
- `uint16_t spi_set_divisor (const uint16_t clkdivider)`

6.10.1 Define Documentation

6.10.1.1 `#define SPI_16MHz 1 /* 16MHz/16Mhz */`

Definition at line 20 of file spi.h.

6.10.1.2 `#define SPI_1MHz 16 /* 16MHz/1MHz */`

Definition at line 16 of file spi.h.

6.10.1.3 **#define SPI_250kHz 64 /* 16MHz/250000 */**

SMCLK divider arguments for `spi_set_divisor` assumes 16MHz SMCLK. You need to change if you use a different frequency

Definition at line 14 of file `spi.h`.

6.10.1.4 **#define SPI_2MHz 8 /* 16MHz/2MHz */**

Definition at line 17 of file `spi.h`.

6.10.1.5 **#define SPI_400kHz 40 /* 16MHz/400000 */**

Definition at line 15 of file `spi.h`.

6.10.1.6 **#define SPI_4MHz 4 /* 16MHz/4MHz */**

Definition at line 18 of file `spi.h`.

6.10.1.7 **#define SPI_8MHz 2 /* 16MHz/8MHz */**

Definition at line 19 of file `spi.h`.

6.10.1.8 **#define SPI_DEFAULT_SPEED SPI_8MHz**

Definition at line 22 of file `spi.h`.

6.10.2 Function Documentation

6.10.2.1 **void spi_initialize (void)**

[`spi_initialize\(\)`](#) - Configure USCI UCB0 for SPI mode

P2.0 - CS (active low) P1.5 - SCLK P1.6 - SIMO/MOSI P1.7 - SOMI/MISO

Definition at line 42 of file `spi.c`.

6.10.2.2 **uint8_t spi_receive (void)**

[`spi_receive\(\)`](#) - send dummy byte then recv response

Definition at line 80 of file `spi.c`.

6.10.2.3 uint8_t spi_send (const uint8_t c)

[spi_send\(\)](#) - send a byte and recv response

Definition at line 64 of file spi.c.

6.10.2.4 uint16_t spi_set_divisor (const uint16_t clkdiv)

[spi_set_divisor\(\)](#) - set new clock divider for USCI

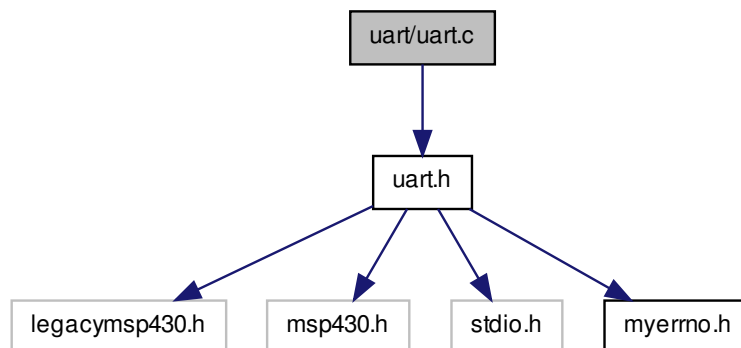
USCI speed is based on the SMCLK divided by BR0 and BR1 initially we start slow (400kHz) to conform to SD Card specifications then we speed up once initialized (SPI_DEFAULT_SPEED)

returns the previous setting

Definition at line 103 of file spi.c.

6.11 uart/uart.c File Reference

`#include <uart.h>` Include dependency graph for uart.c:



Functions

- int [uart_initialize](#) ()
- int [uart_set_birate](#) (unsigned int rate)
- int [uart_close](#) ()
- int [getchar](#) ()
- int [putchar](#) (volatile int c)

- [interrupt](#) (USCIAB0TX_VECTOR)

6.11.1 Function Documentation

6.11.1.1 int getchar ()

Default function that capture a char from the standard input

Returns

Character captured

Definition at line 48 of file uart.c.

6.11.1.2 interrupt (USCIAB0TX_VECTOR)

Interrupr function from TX of USCI 0 It's just commute P1.0 (OUTPUT LED)

Definition at line 73 of file uart.c.

6.11.1.3 int putchar (int *ch*)

Default function that print a char on the standard output

Returns

Return the character written as an unsigned char cast to an int or EOF on error.

Definition at line 55 of file uart.c.

6.11.1.4 int uart_close ()

Close the module of the uart

Returns

Return if the module was full disabled or not 0: Even the specific interrupt as the global interrupt was disabled 1: The global interrupt still enabled

Definition at line 41 of file uart.c.

6.11.1.5 int uart_initialize ()

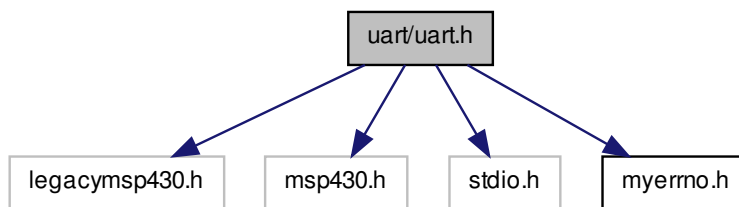
Definition at line 7 of file uart.c.

6.11.1.6 int uart_set_brate (unsigned int rate)

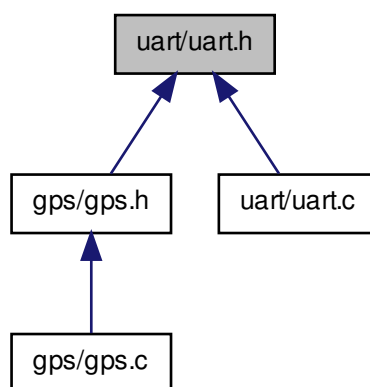
Definition at line 30 of file uart.c.

6.12 uart/uart.h File Reference

```
#include <legacymsp430.h> #include <msp430.h> #include  
<stdio.h> #include <myerrno.h> Include dependency graph for uart-  
h:
```



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



Functions

- int [uart_initialize](#) ()
- int [uart_close](#) ()
- int [getchar](#) ()
- int [putchar](#) (int ch)

6.12.1 Function Documentation

6.12.1.1 int [getchar](#) ()

Default function that capture a char from the standard input

Returns

Character captured

Definition at line 48 of file `uart.c`.

6.12.1.2 int [putchar](#) (int *ch*)

Default function that print a char on the standard output

Returns

Return the character written as an unsigned char cast to an int or EOF on error.

Definition at line 55 of file `uart.c`.

6.12.1.3 int [uart_close](#) ()

Close the module of the uart

Returns

Return if the module was full disabled or not 0: Even the specific interrupt as the global interrupt was disabled 1: The global interrupt still enabled

Definition at line 41 of file `uart.c`.

6.12.1.4 int [uart_initialize](#) ()

Definition at line 7 of file `uart.c`.