

Avrsynth

0.1.1

Generated by Doxygen 1.8.6

Sat Feb 15 2014 14:43:15

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	DIR Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	clust	6
3.1.1.2	dir	6
3.1.1.3	fn	6
3.1.1.4	fs	6
3.1.1.5	id	6
3.1.1.6	index	6
3.1.1.7	sclust	6
3.1.1.8	sect	6
3.2	FATFS Struct Reference	6
3.2.1	Field Documentation	7
3.2.1.1	csize	7
3.2.1.2	database	7
3.2.1.3	dirbase	7
3.2.1.4	drv	7
3.2.1.5	fatbase	7
3.2.1.6	free_clust	7
3.2.1.7	fs_type	7
3.2.1.8	fsi_flag	8
3.2.1.9	fsi_sector	8
3.2.1.10	fsize	8

3.2.1.11	id	8
3.2.1.12	last_clust	8
3.2.1.13	n_fatent	8
3.2.1.14	n_fats	8
3.2.1.15	n_rootdir	8
3.2.1.16	volbase	8
3.2.1.17	wflag	8
3.2.1.18	win	8
3.2.1.19	winsect	9
3.3	FIL Struct Reference	9
3.3.1	Field Documentation	9
3.3.1.1	clust	9
3.3.1.2	dir_ptr	10
3.3.1.3	dir_sect	10
3.3.1.4	dsect	10
3.3.1.5	flag	10
3.3.1.6	fptr	10
3.3.1.7	fs	10
3.3.1.8	fsize	10
3.3.1.9	id	10
3.3.1.10	pad1	10
3.3.1.11	sclust	10
3.4	FILINFO Struct Reference	10
3.4.1	Field Documentation	11
3.4.1.1	fattrib	11
3.4.1.2	fdate	11
3.4.1.3	fname	11
3.4.1.4	fsize	11
3.4.1.5	ftime	11
3.5	onode Struct Reference	11
3.5.1	Field Documentation	12
3.5.1.1	data	12
3.5.1.2	next	12
3.5.1.3	prev	12
3.6	order Struct Reference	12
3.6.1	Field Documentation	12
3.6.1.1	bank1_note	12

3.6.1.2	bank1_startstop	12
3.6.1.3	bank2_note	12
3.6.1.4	bank2_startstop	13
3.6.1.5	bank3_note	13
3.6.1.6	bank3_startstop	13
3.6.1.7	id	13
4	File Documentation	15
4.1	config.h File Reference	15
4.1.1	Macro Definition Documentation	15
4.1.1.1	F_CPU	15
4.2	keyboard/keyboard.c File Reference	16
4.2.1	Macro Definition Documentation	17
4.2.1.1	KB_CLK	17
4.2.1.2	KB_DATA	17
4.2.1.3	KB_PORT	17
4.2.2	Function Documentation	17
4.2.2.1	clock_init	17
4.2.2.2	init_keyboard	17
4.2.2.3	ISR	17
4.2.2.4	main	17
4.2.2.5	read_char	17
4.2.2.6	render_scan_code	17
4.2.3	Variable Documentation	18
4.2.3.1	bit_count	18
4.2.3.2	caps_lock	18
4.2.3.3	char_waiting	18
4.2.3.4	extended	18
4.2.3.5	kbd_data	18
4.2.3.6	release	18
4.2.3.7	shift	18
4.2.3.8	st	18
4.2.3.9	started	18
4.3	keyboard/keymap.h File Reference	19
4.3.1	Variable Documentation	19
4.3.1.1	PROGMEM	19
4.4	lcd8bit/lcd8bit.c File Reference	19

4.4.1	Macro Definition Documentation	20
4.4.1.1	COMM_PORT	20
4.4.1.2	DATA_PORT	20
4.4.1.3	LCD_E	20
4.4.1.4	LCD_RS	20
4.4.2	Function Documentation	20
4.4.2.1	lcd_clear_and_home	20
4.4.2.2	lcd_goto	21
4.4.2.3	lcd_home	21
4.4.2.4	lcd_init	21
4.4.2.5	lcd_line_one	21
4.4.2.6	lcd_line_two	21
4.4.2.7	lcd_set_write_data	21
4.4.2.8	lcd_set_write_instruction	21
4.4.2.9	lcd_write_byte	21
4.4.2.10	lcd_write_data	21
4.4.2.11	lcd_write_string	22
4.4.2.12	lcd_write_string_0	22
4.4.2.13	lcd_write_string_p	22
4.5	lcd8bit/lcd8bit.h File Reference	22
4.5.1	Function Documentation	23
4.5.1.1	lcd_clear_and_home	23
4.5.1.2	lcd_goto	23
4.5.1.3	lcd_home	23
4.5.1.4	lcd_init	23
4.5.1.5	lcd_line_one	23
4.5.1.6	lcd_line_two	23
4.5.1.7	lcd_set_write_data	23
4.5.1.8	lcd_set_write_instruction	23
4.5.1.9	lcd_write_byte	23
4.5.1.10	lcd_write_data	24
4.5.1.11	lcd_write_string	24
4.5.1.12	lcd_write_string_0	24
4.5.1.13	lcd_write_string_p	24
4.6	list/list.c File Reference	24
4.6.1	Function Documentation	25
4.6.1.1	deleteJustList	25

4.6.1.2	deleteList	26
4.6.1.3	deleteNode	26
4.6.1.4	deleteNodeOnly	26
4.6.1.5	evictNode	26
4.6.1.6	getNextOrder	26
4.6.1.7	getOrderData	26
4.6.1.8	getOrderId	26
4.6.1.9	getOrderNode	26
4.6.1.10	getOrderNote	27
4.6.1.11	getOrderStartstop	27
4.6.1.12	getPrevOrder	27
4.6.1.13	insertNode	27
4.6.1.14	newNode	27
4.6.1.15	newNodeByRef	27
4.6.1.16	printList	28
4.6.1.17	pushNode	28
4.6.1.18	setOrderId	28
4.6.1.19	setOrderNote	28
4.6.1.20	setOrderStartstop	28
4.6.1.21	sort	28
4.6.1.22	swap	28
4.6.2	Variable Documentation	28
4.6.2.1	g	28
4.7	list/list.h File Reference	29
4.7.1	Function Documentation	30
4.7.1.1	deleteJustList	30
4.7.1.2	deleteList	30
4.7.1.3	deleteNode	30
4.7.1.4	deleteNodeOnly	31
4.7.1.5	evictNode	31
4.7.1.6	getNextOrder	31
4.7.1.7	getOrderData	31
4.7.1.8	getOrderId	31
4.7.1.9	getOrderNode	31
4.7.1.10	getOrderNote	31
4.7.1.11	getOrderStartstop	32
4.7.1.12	getPrevOrder	32

4.7.1.13	insertNode	32
4.7.1.14	newNode	32
4.7.1.15	newNodeByRef	32
4.7.1.16	printList	32
4.7.1.17	pushNode	33
4.7.1.18	setOrderId	33
4.7.1.19	setOrderNote	33
4.7.1.20	setOrderStartstop	33
4.7.1.21	sort	33
4.7.1.22	swap	33
4.8	main.c File Reference	33
4.8.1	Function Documentation	34
4.8.1.1	die	34
4.8.1.2	get_fattime	34
4.8.1.3	main	34
4.8.2	Variable Documentation	34
4.8.2.1	FatFs	34
4.8.2.2	fp	34
4.9	Makefile File Reference	35
4.9.1	Variable Documentation	35
4.9.1.1	DEVICE	35
4.9.1.2	WI	35
4.10	sd_card/diskio.h File Reference	35
4.10.1	Macro Definition Documentation	37
4.10.1.1	ATA_GET_MODEL	37
4.10.1.2	ATA_GET_REV	37
4.10.1.3	ATA_GET_SN	37
4.10.1.4	CT_BLOCK	37
4.10.1.5	CT_MMC	37
4.10.1.6	CT_SD1	37
4.10.1.7	CT_SD2	37
4.10.1.8	CT_SDC	37
4.10.1.9	CTRL_ERASE_SECTOR	37
4.10.1.10	CTRL_POWER	37
4.10.1.11	CTRL_SYNC	37
4.10.1.12	GET_BLOCK_SIZE	38
4.10.1.13	GET_SECTOR_COUNT	38

4.10.1.14 MMC_GET_CID	38
4.10.1.15 MMC_GET_CSD	38
4.10.1.16 MMC_GET_OCR	38
4.10.1.17 MMC_GET_SDSTAT	38
4.10.1.18 MMC_GET_TYPE	38
4.10.1.19 STA_NODISK	38
4.10.1.20 STA_NOINIT	38
4.10.1.21 STA_PROTECT	38
4.10.2 Typedef Documentation	38
4.10.2.1 DSTATUS	38
4.10.3 Enumeration Type Documentation	38
4.10.3.1 DRESULT	38
4.10.4 Function Documentation	38
4.10.4.1 disk_initialize	39
4.10.4.2 disk_ioctl	39
4.10.4.3 disk_read	39
4.10.4.4 disk_status	39
4.10.4.5 disk_write	39
4.11 sd_card/ff.c File Reference	39
4.11.1 Macro Definition Documentation	42
4.11.1.1 _DF1S	42
4.11.1.2 _EXCVT	43
4.11.1.3 ABORT	43
4.11.1.4 BPB_BkBootSec	43
4.11.1.5 BPB_BytsPerSec	43
4.11.1.6 BPB_ExtFlags	43
4.11.1.7 BPB_FATSz16	43
4.11.1.8 BPB_FATSz32	43
4.11.1.9 BPB_FSInfo	43
4.11.1.10 BPB_FSVer	43
4.11.1.11 BPB_HiddSec	43
4.11.1.12 BPB_Media	43
4.11.1.13 BPB_NumFATs	43
4.11.1.14 BPB_NumHeads	44
4.11.1.15 BPB_RootClus	44
4.11.1.16 BPB_RootEntCnt	44
4.11.1.17 BPB_RsvdSecCnt	44

4.11.1.18 BPB_SecPerClus	44
4.11.1.19 BPB_SecPerTrk	44
4.11.1.20 BPB_TotSec16	44
4.11.1.21 BPB_TotSec32	44
4.11.1.22 BS_55AA	44
4.11.1.23 BS_BootSig	44
4.11.1.24 BS_BootSig32	44
4.11.1.25 BS_DrvNum	44
4.11.1.26 BS_DrvNum32	44
4.11.1.27 BS_FilSysType	44
4.11.1.28 BS_FilSysType32	44
4.11.1.29 BS_jmpBoot	45
4.11.1.30 BS_OEMName	45
4.11.1.31 BS_VolID	45
4.11.1.32 BS_VolID32	45
4.11.1.33 BS_VolLab	45
4.11.1.34 BS_VolLab32	45
4.11.1.35 DDE	45
4.11.1.36 DEF_NAMEBUF	45
4.11.1.37 DIR_Attr	45
4.11.1.38 DIR_CrtDate	45
4.11.1.39 DIR_CrtTime	45
4.11.1.40 DIR_CrtTimeTenth	45
4.11.1.41 DIR_FileSize	45
4.11.1.42 DIR_FstClusHI	45
4.11.1.43 DIR_FstClusLO	45
4.11.1.44 DIR_LstAccDate	46
4.11.1.45 DIR_Name	46
4.11.1.46 DIR_NTres	46
4.11.1.47 DIR_WrtDate	46
4.11.1.48 DIR_WrtTime	46
4.11.1.49 ENTER_FF	46
4.11.1.50 FREE_BUF	46
4.11.1.51 FSI_Free_Count	46
4.11.1.52 FSI_LeadSig	46
4.11.1.53 FSI_Nxt_Free	46
4.11.1.54 FSI_StrucSig	46

4.11.1.55 INIT_BUF	46
4.11.1.56 IsDBCS1	47
4.11.1.57 IsDBCS2	47
4.11.1.58 IsDigit	47
4.11.1.59 IsLower	47
4.11.1.60 IsUpper	47
4.11.1.61 LDIR_Attr	47
4.11.1.62 LDIR_Chksum	47
4.11.1.63 LDIR_FstClusLO	47
4.11.1.64 LDIR_Ord	47
4.11.1.65 LDIR_Type	47
4.11.1.66 LEAVE_FF	47
4.11.1.67 LLE	47
4.11.1.68 MBR_Table	47
4.11.1.69 MIN_FAT16	47
4.11.1.70 MIN_FAT32	47
4.11.1.71 NDDE	48
4.11.1.72 NS	48
4.11.1.73 NS_BODY	48
4.11.1.74 NS_DOT	48
4.11.1.75 NS_EXT	48
4.11.1.76 NS_LAST	48
4.11.1.77 NS_LFN	48
4.11.1.78 NS_LOSS	48
4.11.1.79 SS	48
4.11.1.80 SZ_DIR	48
4.11.1.81 SZ_PTE	48
4.11.2 Function Documentation	49
4.11.2.1 check_fs	49
4.11.2.2 chk_chr	49
4.11.2.3 chk_mounted	49
4.11.2.4 clust2sect	49
4.11.2.5 create_chain	49
4.11.2.6 create_name	49
4.11.2.7 dir_alloc	49
4.11.2.8 dir_find	50
4.11.2.9 dir_next	50

4.11.2.10	dir_read	50
4.11.2.11	dir_register	50
4.11.2.12	dir_sdi	50
4.11.2.13	f_close	50
4.11.2.14	f_getlabel	50
4.11.2.15	f_lseek	51
4.11.2.16	f_mount	51
4.11.2.17	f_open	51
4.11.2.18	f_read	51
4.11.2.19	f_setlabel	51
4.11.2.20	f_sync	51
4.11.2.21	f_write	51
4.11.2.22	follow_path	52
4.11.2.23	get_fat	52
4.11.2.24	ld_clust	52
4.11.2.25	mem_cmp	52
4.11.2.26	mem_cpy	52
4.11.2.27	mem_set	52
4.11.2.28	move_window	52
4.11.2.29	put_fat	52
4.11.2.30	remove_chain	53
4.11.2.31	st_clust	53
4.11.2.32	sync_fs	53
4.11.2.33	sync_window	53
4.11.2.34	validate	53
4.11.3	Variable Documentation	53
4.11.3.1	ExCvt	53
4.11.3.2	FatFs	53
4.11.3.3	Fsid	53
4.12	sd_card/ff.h File Reference	54
4.12.1	Macro Definition Documentation	56
4.12.1.1	_FATFS	56
4.12.1.2	_T	56
4.12.1.3	_TEXT	56
4.12.1.4	AM_ARC	56
4.12.1.5	AM_DIR	56
4.12.1.6	AM_HID	57

4.12.1.7	AM_LFN	57
4.12.1.8	AM_MASK	57
4.12.1.9	AM_RDO	57
4.12.1.10	AM_SYS	57
4.12.1.11	AM_VOL	57
4.12.1.12	CREATE_LINKMAP	57
4.12.1.13	EOF	57
4.12.1.14	f_eof	57
4.12.1.15	f_error	57
4.12.1.16	f_size	57
4.12.1.17	f_tell	57
4.12.1.18	FA_DIRTY	57
4.12.1.19	FA_ERROR	57
4.12.1.20	FA_WRITTEN	57
4.12.1.21	FA_CREATE_ALWAYS	58
4.12.1.22	FA_CREATE_NEW	58
4.12.1.23	FA_OPEN_ALWAYS	58
4.12.1.24	FA_OPEN_EXISTING	58
4.12.1.25	FA_READ	58
4.12.1.26	FA_WRITE	58
4.12.1.27	FS_FAT12	58
4.12.1.28	FS_FAT16	58
4.12.1.29	FS_FAT32	58
4.12.1.30	LD2PD	58
4.12.1.31	LD2PT	58
4.12.1.32	LD_DWORD	58
4.12.1.33	LD_WORD	59
4.12.1.34	ST_DWORD	59
4.12.1.35	ST_WORD	59
4.12.2	Typedef Documentation	59
4.12.2.1	TCHAR	59
4.12.3	Enumeration Type Documentation	59
4.12.3.1	FRESULT	59
4.12.4	Function Documentation	60
4.12.4.1	f_chdir	60
4.12.4.2	f_chdrive	60
4.12.4.3	f_chmod	60

4.12.4.4	f_close	60
4.12.4.5	f_disk	60
4.12.4.6	f_forward	60
4.12.4.7	f_getcwd	60
4.12.4.8	f_getfree	60
4.12.4.9	f_getlabel	60
4.12.4.10	f_gets	60
4.12.4.11	f_lseek	60
4.12.4.12	f_mkdir	60
4.12.4.13	f_mkfs	60
4.12.4.14	f_mount	60
4.12.4.15	f_open	61
4.12.4.16	f_opendir	61
4.12.4.17	f_printf	61
4.12.4.18	f_putc	61
4.12.4.19	f_puts	61
4.12.4.20	f_read	61
4.12.4.21	f_readdir	61
4.12.4.22	f_rename	61
4.12.4.23	f_setlabel	61
4.12.4.24	f_stat	61
4.12.4.25	f_sync	61
4.12.4.26	f_truncate	61
4.12.4.27	f_unlink	62
4.12.4.28	f_utime	62
4.12.4.29	f_write	62
4.12.4.30	get_fatime	62
4.13	sd_card/ffconf.h File Reference	62
4.13.1	Macro Definition Documentation	63
4.13.1.1	_CODE_PAGE	63
4.13.1.2	_FFCONF	63
4.13.1.3	_FS_LOCK	63
4.13.1.4	_FS_MINIMIZE	63
4.13.1.5	_FS_READONLY	63
4.13.1.6	_FS_REENTRANT	63
4.13.1.7	_FS_RPATH	63
4.13.1.8	_FS_TIMEOUT	63

4.13.1.9	_FS_TINY	63
4.13.1.10	_LFN_UNICODE	63
4.13.1.11	_MAX_LFN	63
4.13.1.12	_MAX_SS	64
4.13.1.13	_MULTI_PARTITION	64
4.13.1.14	_SYNC_t	64
4.13.1.15	_USE_ERASE	64
4.13.1.16	_USE_FASTSEEK	64
4.13.1.17	_USE_FORWARD	64
4.13.1.18	_USE_LABEL	64
4.13.1.19	_USE_LFN	64
4.13.1.20	_USE_MKFS	64
4.13.1.21	_USE_STRFUNC	64
4.13.1.22	_VOLUMES	64
4.13.1.23	_WORD_ACCESS	64
4.14	sd_card/integer.h File Reference	64
4.14.1	Typedef Documentation	65
4.14.1.1	BYTE	65
4.14.1.2	CHAR	65
4.14.1.3	DWORD	65
4.14.1.4	INT	65
4.14.1.5	LONG	65
4.14.1.6	SHORT	65
4.14.1.7	UCHAR	65
4.14.1.8	UINT	65
4.14.1.9	ULONG	65
4.14.1.10	USHORT	65
4.14.1.11	WCHAR	65
4.14.1.12	WORD	65
4.15	sd_card/sdmm.c File Reference	65
4.15.1	Macro Definition Documentation	67
4.15.1.1	ACMD13	67
4.15.1.2	ACMD23	67
4.15.1.3	ACMD41	67
4.15.1.4	CK_DQ	67
4.15.1.5	CK_H	67
4.15.1.6	CK_INIT	68

4.15.1.7	CK_L	68
4.15.1.8	CMD0	68
4.15.1.9	CMD1	68
4.15.1.10	CMD10	68
4.15.1.11	CMD12	68
4.15.1.12	CMD13	68
4.15.1.13	CMD16	68
4.15.1.14	CMD17	68
4.15.1.15	CMD18	68
4.15.1.16	CMD23	68
4.15.1.17	CMD24	68
4.15.1.18	CMD25	68
4.15.1.19	CMD32	69
4.15.1.20	CMD33	69
4.15.1.21	CMD38	69
4.15.1.22	CMD55	69
4.15.1.23	CMD58	69
4.15.1.24	CMD8	69
4.15.1.25	CMD9	69
4.15.1.26	CS_DQ	69
4.15.1.27	CS_H	69
4.15.1.28	CS_INIT	69
4.15.1.29	CS_L	69
4.15.1.30	DI_DQ	69
4.15.1.31	DI_H	69
4.15.1.32	DI_INIT	69
4.15.1.33	DI_L	70
4.15.1.34	DO	70
4.15.1.35	DO_DQ	70
4.15.1.36	DO_INIT	70
4.15.1.37	SD_COM_PORT	70
4.15.2	Function Documentation	70
4.15.2.1	deselect	70
4.15.2.2	disk_initialize	70
4.15.2.3	disk_ioctl	70
4.15.2.4	disk_read	70
4.15.2.5	disk_status	71

4.15.2.6	disk_write	71
4.15.2.7	dly_us	71
4.15.2.8	rcvr_datablock	71
4.15.2.9	rcvr_mmc	71
4.15.2.10	select	71
4.15.2.11	send_cmd	71
4.15.2.12	wait_ready	71
4.15.2.13	xmit_datablock	71
4.15.2.14	xmit_mmc	72
4.15.3	Variable Documentation	72
4.15.3.1	CardType	72
4.15.3.2	Stat	72
4.16	sd_card/timeout.h File Reference	72
4.16.1	Macro Definition Documentation	72
4.16.1.1	F_CPU	72
4.17	wavegen/wavegen.c File Reference	73
4.17.1	Function Documentation	74
4.17.1.1	ISR	74
4.17.1.2	ISR	74
4.17.1.3	wavegen_clock_init	74
4.17.1.4	wavegen_disableSound	74
4.17.1.5	wavegen_noiseWave	74
4.17.1.6	wavegen_pwmInit	74
4.17.1.7	wavegen_pwmSet	74
4.17.1.8	wavegen_setFrequency	74
4.17.1.9	wavegen_setFrequency2	74
4.17.1.10	wavegen_setSound	74
4.17.1.11	wavegen_sineWave	75
4.17.2	Variable Documentation	75
4.17.2.1	currentVoice	75
4.17.2.2	frequencyCoef	75
4.17.2.3	frequencyCoef2	75
4.17.2.4	PROGMEM	75
4.17.2.5	sound1Enabled	75
4.17.2.6	sound2Enabled	75
4.17.2.7	sound3Enabled	75
4.17.2.8	soundPWM	76

4.17.2.9 wavetable	76
4.17.2.10 wavetable2	76
4.17.2.11 wavetable3	76
4.18 wavegen/wavegen.h File Reference	76
4.18.1 Macro Definition Documentation	77
4.18.1.1 FINT	77
4.18.1.2 FS	77
4.18.1.3 INTERRUPT_PERIOD	77
4.18.2 Function Documentation	77
4.18.2.1 wavegen_clock_init	77
4.18.2.2 wavegen_disableSound	78
4.18.2.3 wavegen_noiseWave	78
4.18.2.4 wavegen_pwmInit	78
4.18.2.5 wavegen_pwmSet	78
4.18.2.6 wavegen_setFrequency	78
4.18.2.7 wavegen_setFrequency2	78
4.18.2.8 wavegen_setSound	78
4.18.2.9 wavegen_sineWave	78
Index	79

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

DIR	5
FATFS	6
FIL	9
FILINFO	10
onode	11
order	12

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

config.h	15
main.c	33
Makefile	35
keyboard/keyboard.c	16
keyboard/keymap.h	19
lcd8bit/lcd8bit.c	19
lcd8bit/lcd8bit.h	22
list/list.c	24
list/list.h	29
sd_card/diskio.h	35
sd_card/ff.c	39
sd_card/ff.h	54
sd_card/ffconf.h	62
sd_card/integer.h	64
sd_card/sdmm.c	65
sd_card/timeout.h	72
wavegen/wavegen.c	73
wavegen/wavegen.h	76

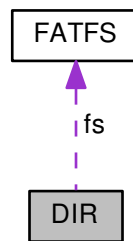
Chapter 3

Data Structure Documentation

3.1 DIR Struct Reference

```
#include <ff.h>
```

Collaboration diagram for DIR:



Data Fields

- [FATFS * fs](#)
- [WORD id](#)
- [WORD index](#)
- [DWORD sclust](#)
- [DWORD clust](#)
- [DWORD sect](#)
- [BYTE * dir](#)
- [BYTE * fn](#)

3.1.1 Field Documentation

3.1.1.1 **DWORD DIR::clust**

Referenced by `dir_next()`, and `dir_sdi()`.

3.1.1.2 **BYTE* DIR::dir**

Referenced by `dir_alloc()`, `dir_find()`, `dir_next()`, `dir_read()`, `dir_register()`, `dir_sdi()`, `f_getlabel()`, `f_open()`, `f_setlabel()`, and `follow_path()`.

3.1.1.3 **BYTE* DIR::fn**

Referenced by `create_name()`, `dir_find()`, `dir_register()`, and `follow_path()`.

3.1.1.4 **FATFS* DIR::fs**

Referenced by `dir_alloc()`, `dir_find()`, `dir_next()`, `dir_read()`, `dir_register()`, `dir_sdi()`, `f_getlabel()`, `f_open()`, `f_setlabel()`, and `follow_path()`.

3.1.1.5 **WORD DIR::id**

3.1.1.6 **WORD DIR::index**

Referenced by `dir_find()`, `dir_next()`, `dir_read()`, `dir_register()`, and `dir_sdi()`.

3.1.1.7 **DWORD DIR::sclust**

Referenced by `dir_sdi()`, `f_getlabel()`, `f_setlabel()`, and `follow_path()`.

3.1.1.8 **DWORD DIR::sect**

Referenced by `dir_alloc()`, `dir_find()`, `dir_next()`, `dir_read()`, `dir_register()`, and `dir_sdi()`.

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

- `sd_card/ff.h`

3.2 FATFS Struct Reference

```
#include <ff.h>
```

Data Fields

- `BYTE fs_type`
- `BYTE drv`
- `BYTE csize`
- `BYTE n_fats`
- `BYTE wflag`

- [BYTE fsi_flag](#)
- [WORD id](#)
- [WORD n_rootdir](#)
- [DWORD last_clust](#)
- [DWORD free_clust](#)
- [DWORD fsi_sector](#)
- [DWORD n_fatent](#)
- [DWORD fsize](#)
- [DWORD volbase](#)
- [DWORD fatbase](#)
- [DWORD dirbase](#)
- [DWORD database](#)
- [DWORD winsect](#)
- [BYTE win](#) [[_MAX_SS](#)]

3.2.1 Field Documentation

3.2.1.1 BYTE FATFS::csize

Referenced by [chk_mounted\(\)](#), [clust2sect\(\)](#), [dir_next\(\)](#), [dir_sdi\(\)](#), [f_lseek\(\)](#), [f_read\(\)](#), [f_write\(\)](#), and [remove_chain\(\)](#).

3.2.1.2 DWORD FATFS::database

Referenced by [chk_mounted\(\)](#), and [clust2sect\(\)](#).

3.2.1.3 DWORD FATFS::dirbase

Referenced by [chk_mounted\(\)](#), and [dir_sdi\(\)](#).

3.2.1.4 BYTE FATFS::drv

Referenced by [check_fs\(\)](#), [chk_mounted\(\)](#), [f_lseek\(\)](#), [f_read\(\)](#), [f_sync\(\)](#), [f_write\(\)](#), [move_window\(\)](#), [remove_chain\(\)](#), [sync_fs\(\)](#), [sync_window\(\)](#), and [validate\(\)](#).

3.2.1.5 DWORD FATFS::fatbase

Referenced by [chk_mounted\(\)](#), [get_fat\(\)](#), [put_fat\(\)](#), and [sync_window\(\)](#).

3.2.1.6 DWORD FATFS::free_clust

Referenced by [chk_mounted\(\)](#), [create_chain\(\)](#), [remove_chain\(\)](#), and [sync_fs\(\)](#).

3.2.1.7 BYTE FATFS::fs_type

Referenced by [chk_mounted\(\)](#), [dir_sdi\(\)](#), [f_getlabel\(\)](#), [f_mount\(\)](#), [get_fat\(\)](#), [ld_clust\(\)](#), [put_fat\(\)](#), [sync_fs\(\)](#), and [validate\(\)](#).

3.2.1.8 BYTE FATFS::fsi_flag

Referenced by `chk_mounted()`, `create_chain()`, `remove_chain()`, and `sync_fs()`.

3.2.1.9 DWORD FATFS::fsi_sector

Referenced by `chk_mounted()`, and `sync_fs()`.

3.2.1.10 DWORD FATFS::fsize

Referenced by `chk_mounted()`, and `sync_window()`.

3.2.1.11 WORD FATFS::id

Referenced by `chk_mounted()`, `f_open()`, and `validate()`.

3.2.1.12 DWORD FATFS::last_clust

Referenced by `chk_mounted()`, `create_chain()`, `f_open()`, and `sync_fs()`.

3.2.1.13 DWORD FATFS::n_fatent

Referenced by `chk_mounted()`, `clust2sect()`, `create_chain()`, `dir_next()`, `dir_sdi()`, `f_lseek()`, `get_fat()`, `put_fat()`, and `remove_chain()`.

3.2.1.14 BYTE FATFS::n_fats

Referenced by `chk_mounted()`, and `sync_window()`.

3.2.1.15 WORD FATFS::n_rootdir

Referenced by `chk_mounted()`, `dir_next()`, and `dir_sdi()`.

3.2.1.16 DWORD FATFS::volbase

Referenced by `chk_mounted()`, and `f_getlabel()`.

3.2.1.17 BYTE FATFS::wflag

Referenced by `chk_mounted()`, `dir_next()`, `dir_register()`, `f_open()`, `f_read()`, `f_setlabel()`, `f_sync()`, `f_write()`, `put_fat()`, and `sync_window()`.

3.2.1.18 BYTE FATFS::win[_MAX_SS]

Referenced by `check_fs()`, `chk_mounted()`, `dir_next()`, `dir_sdi()`, `f_getlabel()`, `f_read()`, `f_write()`, `get_fat()`, `move_window()`, `put_fat()`, `sync_fs()`, and `sync_window()`.

3.2.1.19 DWORD FATFS::winsect

Referenced by `chk_mounted()`, `dir_next()`, `f_open()`, `f_read()`, `f_write()`, `move_window()`, `sync_fs()`, and `sync_window()`.

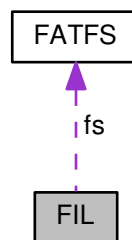
The documentation for this struct was generated from the following file:

- `sd_card/ff.h`

3.3 FIL Struct Reference

```
#include <ff.h>
```

Collaboration diagram for FIL:



Data Fields

- `FATFS * fs`
- `WORD id`
- `BYTE flag`
- `BYTE pad1`
- `DWORD fptr`
- `DWORD fsize`
- `DWORD sclust`
- `DWORD clust`
- `DWORD dsect`
- `DWORD dir_sect`
- `BYTE * dir_ptr`

3.3.1 Field Documentation

3.3.1.1 DWORD FIL::clust

Referenced by `f_lseek()`, `f_read()`, and `f_write()`.

3.3.1.2 **BYTE** FIL::dir_ptr

Referenced by `f_open()`, and `f_sync()`.

3.3.1.3 **DWORD** FIL::dir_sect

Referenced by `f_open()`, and `f_sync()`.

3.3.1.4 **DWORD** FIL::dsect

Referenced by `f_lseek()`, `f_open()`, `f_read()`, `f_sync()`, and `f_write()`.

3.3.1.5 **BYTE** FIL::flag

Referenced by `f_lseek()`, `f_open()`, `f_read()`, `f_sync()`, and `f_write()`.

3.3.1.6 **DWORD** FIL::fptr

Referenced by `f_lseek()`, `f_open()`, `f_read()`, and `f_write()`.

3.3.1.7 **FATFS*** FIL::fs

Referenced by `f_close()`, `f_lseek()`, `f_open()`, `f_read()`, `f_sync()`, `f_write()`, and `validate()`.

3.3.1.8 **DWORD** FIL::fsize

Referenced by `f_lseek()`, `f_open()`, `f_read()`, `f_sync()`, and `f_write()`.

3.3.1.9 **WORD** FIL::id

Referenced by `f_open()`, and `validate()`.

3.3.1.10 **BYTE** FIL::pad1

3.3.1.11 **DWORD** FIL::sclust

Referenced by `f_lseek()`, `f_open()`, `f_read()`, `f_sync()`, and `f_write()`.

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

- `sd_card/ff.h`

3.4 **FILINFO** Struct Reference

```
#include <ff.h>
```

Data Fields

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

3.4.1 Field Documentation

3.4.1.1 BYTE FILINFO::fattrib

3.4.1.2 WORD FILINFO::fdate

3.4.1.3 TCHAR FILINFO::fname[13]

3.4.1.4 DWORD FILINFO::fsize

3.4.1.5 WORD FILINFO::ftime

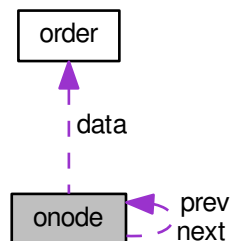
The documentation for this struct was generated from the following file:

- [sd_card/ff.h](#)

3.5 onode Struct Reference

```
#include <list.h>
```

Collaboration diagram for onode:



Data Fields

- struct [order](#) * [data](#)
- struct [onode](#) * [next](#)
- struct [onode](#) * [prev](#)

3.5.1 Field Documentation

3.5.1.1 struct order* onode::data

Referenced by deleteList(), deleteNode(), getOrderData(), getOrderNode(), main(), newNodeByRef(), printList(), sort(), and swap().

3.5.1.2 struct onode* onode::next

Referenced by deleteJustList(), deleteList(), evictNode(), getNextOrder(), getOrderNode(), insertNode(), printList(), and pushNode().

3.5.1.3 struct onode* onode::prev

Referenced by evictNode(), getPrevOrder(), insertNode(), and pushNode().

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

- [list/list.h](#)

3.6 order Struct Reference

```
#include <list.h>
```

Data Fields

- int [id](#)
- char [bank1_startstop](#)
- char [bank1_note](#)
- char [bank2_startstop](#)
- char [bank2_note](#)
- char [bank3_startstop](#)
- char [bank3_note](#)

3.6.1 Field Documentation

3.6.1.1 char order::bank1_note

Referenced by getOrderNote(), main(), newNode(), and setOrderNote().

3.6.1.2 char order::bank1_startstop

Referenced by getOrderStartstop(), main(), newNode(), and setOrderStartstop().

3.6.1.3 char order::bank2_note

Referenced by getOrderNote(), main(), newNode(), and setOrderNote().

3.6.1.4 char order::bank2_startstop

Referenced by getOrderStartstop(), main(), newNode(), and setOrderStartstop().

3.6.1.5 char order::bank3_note

Referenced by getOrderNote(), main(), newNode(), and setOrderNote().

3.6.1.6 char order::bank3_startstop

Referenced by getOrderStartstop(), main(), newNode(), and setOrderStartstop().

3.6.1.7 int order::id

Referenced by getOrderId(), getOrderNode(), main(), newNode(), and setOrderId().

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

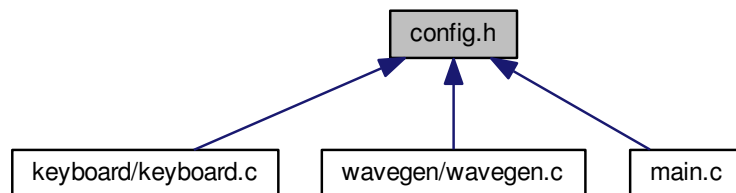
- [list/list.h](#)

Chapter 4

File Documentation

4.1 config.h File Reference

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



Macros

- `#define F_CPU 32000000UL`

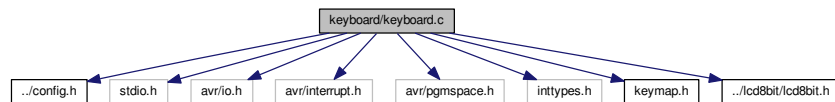
4.1.1 Macro Definition Documentation

4.1.1.1 `#define F_CPU 32000000UL`

4.2 keyboard/keyboard.c File Reference

```
#include "../config.h"
#include <stdio.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <inttypes.h>
#include "keymap.h"
#include "../lcd8bit/lcd8bit.h"
```

Include dependency graph for keyboard.c:



Macros

- `#define KB_CLK PIN2_bm`
- `#define KB_DATA PIN1_bm`
- `#define KB_PORT PORTF`

Functions

- `ISR (PORTF_INT1_vect)`
- `void clock_init (void)`
- `char render_scan_code (uint8_t data)`
- `uint8_t read_char ()`
- `void init_keyboard ()`
- `int main ()`

Variables

- `char st [20] = " "`
- `volatile uint8_t kbd_data`
- `volatile uint8_t char_waiting`
- `uint8_t started`
- `uint8_t bit_count`
- `uint8_t shift`
- `uint8_t caps_lock`
- `uint8_t extended`
- `uint8_t release`

4.2.1 Macro Definition Documentation

4.2.1.1 `#define KB_CLK PIN2_bm`

Referenced by `init_keyboard()`, and `ISR()`.

4.2.1.2 `#define KB_DATA PIN1_bm`

Referenced by `init_keyboard()`, and `ISR()`.

4.2.1.3 `#define KB_PORT PORTF`

Referenced by `init_keyboard()`, and `ISR()`.

4.2.2 Function Documentation

4.2.2.1 `void clock_init (void)`

Referenced by `main()`.

4.2.2.2 `void init_keyboard ()`

References `bit_count`, `KB_CLK`, `KB_DATA`, `KB_PORT`, `kbd_data`, and `started`.

Referenced by `main()`.

4.2.2.3 `ISR (PORTF_INT1_vect)`

References `bit_count`, `char_waiting`, `KB_CLK`, `KB_DATA`, `KB_PORT`, `kbd_data`, `release`, `shift`, and `started`.

4.2.2.4 `int main ()`

References `char_waiting`, `clock_init()`, `init_keyboard()`, `lcd_clear_and_home()`, `lcd_init()`, `lcd_line_one()`, `lcd_line_two()`, `lcd_write_string_0()`, `read_char()`, `render_scan_code()`, and `shift`.

4.2.2.5 `uint8_t read_char ()`

References `char_waiting`, and `kbd_data`.

Referenced by `main()`.

4.2.2.6 `char render_scan_code (uint8_t data)`

References `shift`.

Referenced by `main()`.

4.2.3 Variable Documentation

4.2.3.1 `uint8_t bit_count`

Referenced by `init_keyboard()`, and `ISR()`.

4.2.3.2 `uint8_t caps_lock`

4.2.3.3 `volatile uint8_t char_waiting`

Referenced by `ISR()`, `main()`, and `read_char()`.

4.2.3.4 `uint8_t extended`

4.2.3.5 `volatile uint8_t kbd_data`

Referenced by `init_keyboard()`, `ISR()`, and `read_char()`.

4.2.3.6 `uint8_t release`

Referenced by `ISR()`.

4.2.3.7 `uint8_t shift`

Referenced by `ISR()`, `main()`, and `render_scan_code()`.

4.2.3.8 `char st[20] = " "`

4.2.3.9 `uint8_t started`

Referenced by `init_keyboard()`, and `ISR()`.

Macros

- `#define DATA_PORT PORTB`
- `#define LCD_RS PIN0_bm /* RS on pin PB3 */`
- `#define LCD_E PIN1_bm /* E on pin PB1 */`
- `#define COMM_PORT PORTA`

Functions

- void `lcd_set_write_instruction` ()
- void `lcd_set_write_data` ()
- void `lcd_write_byte` (char c)
- void `lcd_clear_and_home` ()
- void `lcd_home` ()
- void `lcd_goto` (uint8_t line, uint8_t pos)
- void `lcd_line_one` ()
- void `lcd_line_two` ()
- void `lcd_write_data` (char c)
- void `lcd_write_string` (char *x, uint8_t len)
- void `lcd_write_string_0` (char *x)
- void `lcd_write_string_p` (const char *s)
- void `lcd_init` ()

4.4.1 Macro Definition Documentation

4.4.1.1 `#define COMM_PORT PORTA`

Referenced by `lcd_init()`, `lcd_set_write_data()`, `lcd_set_write_instruction()`, and `lcd_write_byte()`.

4.4.1.2 `#define DATA_PORT PORTB`

Referenced by `lcd_init()`, and `lcd_write_byte()`.

4.4.1.3 `#define LCD_E PIN1_bm /* E on pin PB1 */`

Referenced by `lcd_init()`, and `lcd_write_byte()`.

4.4.1.4 `#define LCD_RS PIN0_bm /* RS on pin PB3 */`

Referenced by `lcd_init()`, `lcd_set_write_data()`, and `lcd_set_write_instruction()`.

4.4.2 Function Documentation

4.4.2.1 void `lcd_clear_and_home` ()

References `lcd_set_write_instruction()`, and `lcd_write_byte()`.

Referenced by `die()`, `lcd_init()`, and `main()`.

4.4.2.2 void lcd_goto (uint8_t *line*, uint8_t *pos*)

References lcd_set_write_instruction(), and lcd_write_byte().

Referenced by lcd_line_one(), and lcd_line_two().

4.4.2.3 void lcd_home ()

References lcd_set_write_instruction(), and lcd_write_byte().

4.4.2.4 void lcd_init ()

References COMM_PORT, DATA_PORT, lcd_clear_and_home(), LCD_E, LCD_RS, and lcd_write_byte().

Referenced by main().

4.4.2.5 void lcd_line_one ()

References lcd_goto().

Referenced by die(), and main().

4.4.2.6 void lcd_line_two ()

References lcd_goto().

Referenced by die(), and main().

4.4.2.7 void lcd_set_write_data ()

References COMM_PORT, and LCD_RS.

Referenced by lcd_write_data().

4.4.2.8 void lcd_set_write_instruction ()

References COMM_PORT, and LCD_RS.

Referenced by lcd_clear_and_home(), lcd_goto(), and lcd_home().

4.4.2.9 void lcd_write_byte (char *c*)

References COMM_PORT, DATA_PORT, and LCD_E.

Referenced by lcd_clear_and_home(), lcd_goto(), lcd_home(), lcd_init(), and lcd_write_data().

4.4.2.10 void lcd_write_data (char *c*)

References lcd_set_write_data(), and lcd_write_byte().

Referenced by lcd_write_string(), lcd_write_string_0(), and lcd_write_string_p().

4.4.2.11 `void lcd_write_string (char * x, uint8_t len)`

References `lcd_write_data()`.

4.4.2.12 `void lcd_write_string_0 (char * x)`

References `lcd_write_data()`.

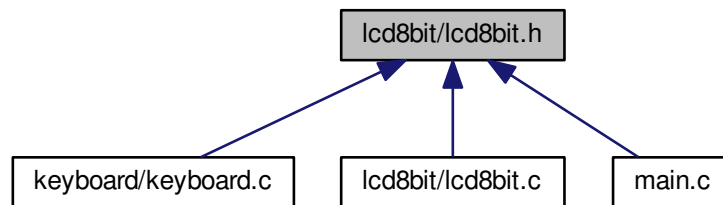
Referenced by `die()`, and `main()`.

4.4.2.13 `void lcd_write_string_p (const char * s)`

References `lcd_write_data()`.

4.5 lcd8bit/lcd8bit.h File Reference

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



Functions

- void `lcd_set_write_instruction ()`
- void `lcd_set_write_data ()`
- void `lcd_write_byte (char c)`
- void `lcd_clear_and_home ()`
- void `lcd_home ()`
- void `lcd_goto (uint8_t line, uint8_t pos)`
- void `lcd_line_one ()`
- void `lcd_line_two ()`
- void `lcd_write_data (char c)`
- void `lcd_write_string (char *x, uint8_t len)`
- void `lcd_write_string_0 (char *x)`
- void `lcd_write_string_p (const char *s)`
- void `lcd_init ()`

4.5.1 Function Documentation

4.5.1.1 void lcd_clear_and_home ()

References lcd_set_write_instruction(), and lcd_write_byte().

Referenced by die(), lcd_init(), and main().

4.5.1.2 void lcd_goto (uint8_t *line*, uint8_t *pos*)

References lcd_set_write_instruction(), and lcd_write_byte().

Referenced by lcd_line_one(), and lcd_line_two().

4.5.1.3 void lcd_home ()

References lcd_set_write_instruction(), and lcd_write_byte().

4.5.1.4 void lcd_init ()

References COMM_PORT, DATA_PORT, lcd_clear_and_home(), LCD_E, LCD_RS, and lcd_write_byte().

Referenced by main().

4.5.1.5 void lcd_line_one ()

References lcd_goto().

Referenced by die(), and main().

4.5.1.6 void lcd_line_two ()

References lcd_goto().

Referenced by die(), and main().

4.5.1.7 void lcd_set_write_data ()

References COMM_PORT, and LCD_RS.

Referenced by lcd_write_data().

4.5.1.8 void lcd_set_write_instruction ()

References COMM_PORT, and LCD_RS.

Referenced by lcd_clear_and_home(), lcd_goto(), and lcd_home().

4.5.1.9 void lcd_write_byte (char *c*)

References COMM_PORT, DATA_PORT, and LCD_E.

Referenced by `lcd_clear_and_home()`, `lcd_goto()`, `lcd_home()`, `lcd_init()`, and `lcd_write_data()`.

4.5.1.10 `void lcd_write_data (char c)`

References `lcd_set_write_data()`, and `lcd_write_byte()`.

Referenced by `lcd_write_string()`, `lcd_write_string_0()`, and `lcd_write_string_p()`.

4.5.1.11 `void lcd_write_string (char * x, uint8_t len)`

References `lcd_write_data()`.

4.5.1.12 `void lcd_write_string_0 (char * x)`

References `lcd_write_data()`.

Referenced by `die()`, and `main()`.

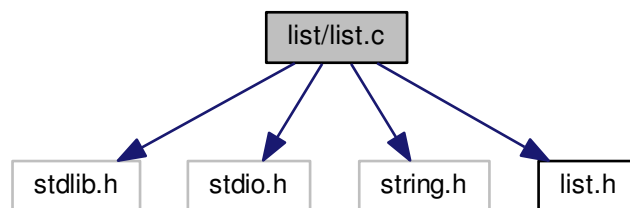
4.5.1.13 `void lcd_write_string_p (const char * s)`

References `lcd_write_data()`.

4.6 list/list.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "list.h"
```

Include dependency graph for `list.c`:



Functions

- `struct onode * newNode (struct order *data)`

Returns a new linked list node filled in with the given order.

- struct `onode` * `newNodeByRef` (struct `order` *data)
Returns a new linked list node filled in with the given order.
- void `pushNode` (struct `onode` **head, struct `onode` *node)
*In a linked list with *head as the head pointer, adds the given node to the front of the list.*
- void `sort` (struct `onode` **head)
Performs insertion sort on the list by comparing the OrderID.
- void `swap` (struct `onode` **head, struct `onode` *node1, struct `onode` *node2)
- void `insertNode` (struct `onode` **head, struct `onode` *prevNode, struct `onode` *insertingNode)
Insert the given node after the prevNode.
- void `evictNode` (struct `onode` **head, struct `onode` *node)
Internally removes the specified node from the list, and updates the linked list.
- void `deleteNodeOnly` (struct `onode` **head, struct `onode` *node)
Removes the specified node from the list, and does NOT free memory the node is using.
- void `deleteNode` (struct `onode` **head, struct `onode` *node)
Removes the specified node from the list, and frees all memory the node is using.
- void `printList` (struct `onode` *node, void(*printItem)(struct `order` *, FILE *), FILE *out)
- struct `onode` * `getNextOrder` (struct `onode` *order_node)
Simply returns the next node in the list, or NULL if there are no further nodes.
- struct `onode` * `getPrevOrder` (struct `onode` *order_node)
- struct `order` * `getOrderData` (struct `onode` *order_node)
- int `getOrderId` (struct `order` *orderData)
- char `getOrderStartstop` (struct `order` *orderData, char bankNum)
- int `getOrderNote` (struct `order` *orderData, char bankNum)
- void `setOrderStartstop` (struct `order` *orderData, char newStartstop, char bankNum)
- void `setOrderNote` (struct `order` *orderData, char newNote, char bankNum)
- void `setOrderId` (struct `order` *orderData, int id)
- void `deleteList` (struct `onode` **head)
*Deletes every node in the list with *head as the head pointer.*
- void `deleteJustList` (struct `onode` **head)
*Deletes every node EXCEPT the head in the list with *head as the head pointer.*

group1 Content Management: Setters and Getters for linkedList

Setters and Getters for linkedList

- struct `onode` * `getOrderNode` (struct `onode` *head, int id)
Returns the pointer to the node containing the given word in the linked list with head as the head pointer.

Variables

- int `g` =0

4.6.1 Function Documentation

4.6.1.1 void deleteJustList (struct `onode` ** head)

Deletes every node EXCEPT the head in the list with *head as the head pointer.

After calling this function, all memory used by the list should be freed, and *head should be NULL.

References `onode::next`.

4.6.1.2 void deleteList (struct onode ** head)

Deletes every node in the list with *head as the head pointer.

After calling this function, all memory used by the list should be freed, and *head should be NULL.

References onode::data, and onode::next.

4.6.1.3 void deleteNode (struct onode ** head, struct onode * node)

Removes the specified node from the list, and frees all memory the node is using.

Remember if *head is the node being deleted, it must be updated.

References onode::data, and deleteNodeOnly().

4.6.1.4 void deleteNodeOnly (struct onode ** head, struct onode * node)

Removes the specified node from the list, and does NOT free memory the node is using.

Remember if *head is the node being deleted, it must be updated.

References evictNode().

Referenced by deleteNode().

4.6.1.5 void evictNode (struct onode ** head, struct onode * node)

Internally removes the specified node from the list, and updates the linked list.

References onode::next, and onode::prev.

Referenced by deleteNodeOnly().

4.6.1.6 struct onode* getNextOrder (struct onode * order_node)

Simply returns the next node in the list, or NULL if there are no further nodes.

References onode::next.

Referenced by sort().

4.6.1.7 struct order* getOrderData (struct onode * order_node)

References onode::data.

4.6.1.8 int getOrderId (struct order * orderData)

References order::id.

Referenced by sort().

4.6.1.9 struct onode * getOrderNode (struct onode * head, int id)

Returns the pointer to the node containing the given word in the linked list with head as the head pointer.

In a linked list with `*head` as the head pointer, returns the onode with the given order id.

If a node with the given word cannot be found, the function returns NULL.

References `onode::data`, `order::id`, and `onode::next`.

Referenced by `main()`.

4.6.1.10 `int getOrderNote (struct order * orderData, char bankNum)`

References `order::bank1_note`, `order::bank2_note`, and `order::bank3_note`.

Referenced by `main()`.

4.6.1.11 `char getOrderStartstop (struct order * orderData, char bankNum)`

References `order::bank1_startstop`, `order::bank2_startstop`, and `order::bank3_startstop`.

Referenced by `main()`.

4.6.1.12 `struct onode* getPrevOrder (struct onode * order_node)`

References `onode::prev`.

Referenced by `sort()`.

4.6.1.13 `void insertNode (struct onode ** head, struct onode * prevNode, struct onode * insertingNode)`

Insert the given node after the `prevNode`.

If the `prevNode` is NULL, then the given node is inserted at the head of the list.

References `onode::next`, `onode::prev`, and `pushNode()`.

4.6.1.14 `struct onode* newNode (struct order * data)`

Returns a new linked list node filled in with the given order.

Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in `data` then allocate a linked list node.

The order is copied into the node as a new instance.

References `order::bank1_note`, `order::bank1_startstop`, `order::bank2_note`, `order::bank2_startstop`, `order::bank3_note`, `order::bank3_startstop`, `g`, and `order::id`.

Referenced by `main()`.

4.6.1.15 `struct onode* newNodeByRef (struct order * data)`

Returns a new linked list node filled in with the given order.

Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in `data` then allocate a linked list node.

The order is inserted by reference.

References `onode::data`.

4.6.1.16 void printList (struct onode * *node*, void(*) (struct order *, FILE *) *printItem*, FILE * *out*)

References onode::data, and onode::next.

4.6.1.17 void pushNode (struct onode ** *head*, struct onode * *node*)

In a linked list with *head as the head pointer, adds the given node to the front of the list.

References onode::next, and onode::prev.

Referenced by insertNode(), and main().

4.6.1.18 void setOrderId (struct order * *orderData*, int *id*)

References order::id.

4.6.1.19 void setOrderNote (struct order * *orderData*, char *newNote*, char *bankNum*)

References order::bank1_note, order::bank2_note, and order::bank3_note.

4.6.1.20 void setOrderStartstop (struct order * *orderData*, char *newStartstop*, char *bankNum*)

References order::bank1_startstop, order::bank2_startstop, and order::bank3_startstop.

4.6.1.21 void sort (struct onode ** *head*)

Performs insertion sort on the list by comparing the OrderID.

Sorts from largest(head) to smallest.

References onode::data, getNextOrder(), getOrderId(), getPrevOrder(), and swap().

Referenced by main().

4.6.1.22 void swap (struct onode ** *head*, struct onode * *node1*, struct onode * *node2*)

References onode::data.

Referenced by sort().

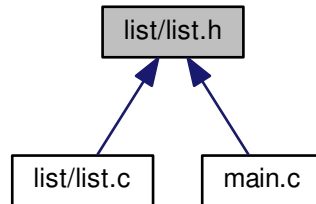
4.6.2 Variable Documentation

4.6.2.1 int g =0

Referenced by newNode().

4.7 list/list.h File Reference

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



Data Structures

- struct [order](#)
- struct [onode](#)

Functions

- struct [onode](#) * [newNode](#) (struct [order](#) *data)
Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in data then allocate a linked list node.
- struct [onode](#) * [newNodeByRef](#) (struct [order](#) *data)
Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in data then allocate a linked list node.
- void [pushNode](#) (struct [onode](#) **head, struct [onode](#) *node)
*In a linked list with *head as the head pointer, adds the given node to the front of the list.*
- struct [onode](#) * [getOrderNode](#) (struct [onode](#) *head, int id)
*In a linked list with *head as the head pointer, returns the onode with the given order id.*
- void [insertNode](#) (struct [onode](#) **head, struct [onode](#) *prevNode, struct [onode](#) *insertingNode)
Insert the given node after the prevNode.
- void [evictNode](#) (struct [onode](#) **head, struct [onode](#) *node)
Internally removes the specified node from the list, and updates the linked list.
- void [deleteNode](#) (struct [onode](#) **head, struct [onode](#) *node)
Removes the specified node from the list, and frees all memory the node is using.
- void [deleteNodeOnly](#) (struct [onode](#) **head, struct [onode](#) *node)
Removes the specified node from the list, and does NOT free memory the node is using.
- void [deleteList](#) (struct [onode](#) **head)
*Deletes every node in the list with *head as the head pointer.*
- void [deleteJustList](#) (struct [onode](#) **head)
*Deletes every node EXCEPT the head in the list with *head as the head pointer.*
- void [printList](#) (struct [onode](#) *node, void(*printItem)(struct [order](#) *, FILE *), FILE *out)

group1 Content Management: Setters and Getters for linkedList

Setters and Getters for linkedList

- struct `onode` * `getNextOrder` (struct `onode` *order_node)
Simply returns the next node in the list, or NULL if there are no further nodes.
- struct `onode` * `getPrevOrder` (struct `onode` *order_node)
- struct `order` * `getOrderData` (struct `onode` *order_node)

group2 Content Management: Setters and Getters for order data

Setters and Getters for linkedList

- int `getOrderId` (struct `order` *orderData)
- char `getOrderStartstop` (struct `order` *orderData, char bankNum)
- int `getOrderNote` (struct `order` *orderData, char bankNum)
- void `setOrderId` (struct `order` *orderData, int id)
- void `setOrderStartstop` (struct `order` *orderData, char newStartstop, char bankNum)
- void `setOrderNote` (struct `order` *orderData, char newNote, char bankNum)

group3 Sorting functions

- void `swap` (struct `onode` **head, struct `onode` *n1, struct `onode` *n2)
- void `sort` (struct `onode` **head)

Performs insertion sort on the list by comparing the OrderID.

4.7.1 Function Documentation

4.7.1.1 void deleteJustList (struct `onode` ** head)

Deletes every node EXCEPT the head in the list with *head as the head pointer.

After calling this function, all memory used by the list should be freed, and *head should be NULL.

References `onode::next`.

4.7.1.2 void deleteList (struct `onode` ** head)

Deletes every node in the list with *head as the head pointer.

After calling this function, all memory used by the list should be freed, and *head should be NULL.

References `onode::data`, and `onode::next`.

4.7.1.3 void deleteNode (struct `onode` ** head, struct `onode` * node)

Removes the specified node from the list, and frees all memory the node is using.

Remember if *head is the node being deleted, it must be updated.

References `onode::data`, and `deleteNodeOnly()`.

4.7.1.4 void deleteNodeOnly (struct onode ** head, struct onode * node)

Removes the specified node from the list, and does NOT free memory the node is using.

Remember if *head is the node being deleted, it must be updated.

References evictNode().

Referenced by deleteNode().

4.7.1.5 void evictNode (struct onode ** head, struct onode * node)

Internally removes the specified node from the list, and updates the linked list.

References onode::next, and onode::prev.

Referenced by deleteNodeOnly().

4.7.1.6 struct onode* getNextOrder (struct onode * order_node)

Simply returns the next node in the list, or NULL if there are no further nodes.

References onode::next.

Referenced by sort().

4.7.1.7 struct order* getOrderData (struct onode * order_node)

References onode::data.

4.7.1.8 int getOrderId (struct order * orderData)

References order::id.

Referenced by sort().

4.7.1.9 struct onode* getOrderNode (struct onode * head, int id)

In a linked list with *head as the head pointer, returns the onode with the given order id.

In a linked list with *head as the head pointer, returns the onode with the given order id.

If a node with the given word cannot be found, the function returns NULL.

References onode::data, order::id, and onode::next.

Referenced by main().

4.7.1.10 int getOrderNote (struct order * orderData, char bankNum)

References order::bank1_note, order::bank2_note, and order::bank3_note.

Referenced by main().

4.7.1.11 `char getOrderStartstop (struct order * orderData, char bankNum)`

References `order::bank1_startstop`, `order::bank2_startstop`, and `order::bank3_startstop`.

Referenced by `main()`.

4.7.1.12 `struct onode* getPrevOrder (struct onode * order_node)`

References `onode::prev`.

Referenced by `sort()`.

4.7.1.13 `void insertNode (struct onode ** head, struct onode * prevNode, struct onode * insertingNode)`

Insert the given node after the `prevNode`.

If the `prevNode` is NULL, then the given node is inserted at the head of the list.

References `onode::next`, `onode::prev`, and `pushNode()`.

4.7.1.14 `struct onode* newNode (struct order * data)`

Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in `data` then allocate a linked list node.

If you are implementing this function make sure that you duplicate, as the original `data` may be modified by the calling function.

Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in `data` then allocate a linked list node.

The order is copied into the node as a new instance.

References `order::bank1_note`, `order::bank1_startstop`, `order::bank2_note`, `order::bank2_startstop`, `order::bank3_note`, `order::bank3_startstop`, `g`, and `order::id`.

Referenced by `main()`.

4.7.1.15 `struct onode* newNodeByRef (struct order * data)`

Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in `data` then allocate a linked list node.

This function is different that `newNode` in that the new node is created by reference and not copied

Returns a new linked list node filled in with the given order, The function allocates a new order and copy the values stored in `data` then allocate a linked list node.

The order is inserted by reference.

References `onode::data`.

4.7.1.16 `void printList (struct onode * node, void(*)(struct order *, FILE *) printItem, FILE * out)`

References `onode::data`, and `onode::next`.

4.7.1.17 void pushNode (struct onode ** head, struct onode * node)

In a linked list with *head as the head pointer, adds the given node to the front of the list.

References onode::next, and onode::prev.

Referenced by insertNode(), and main().

4.7.1.18 void setOrderId (struct order * orderData, int id)

References order::id.

4.7.1.19 void setOrderNote (struct order * orderData, char newNote, char bankNum)

References order::bank1_note, order::bank2_note, and order::bank3_note.

4.7.1.20 void setOrderStartstop (struct order * orderData, char newStartstop, char bankNum)

References order::bank1_startstop, order::bank2_startstop, and order::bank3_startstop.

4.7.1.21 void sort (struct onode ** head)

Performs insertion sort on the list by comparing the OrderID.

Sorts from largest(head) to smallest.

References onode::data, getNextOrder(), getOrderId(), getPrevOrder(), and swap().

Referenced by main().

4.7.1.22 void swap (struct onode ** head, struct onode * n1, struct onode * n2)

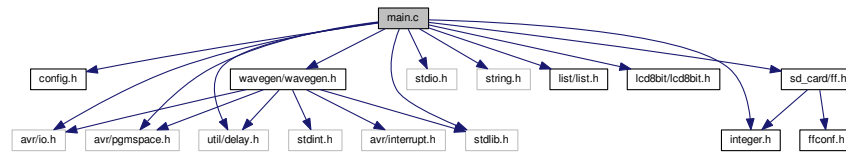
References onode::data.

Referenced by sort().

4.8 main.c File Reference

```
#include "config.h"
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "list/list.h"
#include "lcd8bit/lcd8bit.h"
#include "sd_card/ff.h"
#include "sd_card/integer.h"
#include "wavegen/wavegen.h"
```

Include dependency graph for main.c:



Functions

- [DWORD get_fattime](#) (void)
- void [die](#) (char *message)
- int [main](#) ()

Variables

- [FATFS FatFs](#)
- [FIL * fp](#)

4.8.1 Function Documentation

4.8.1.1 void die (char * message)

References [lcd_clear_and_home\(\)](#), [lcd_line_one\(\)](#), [lcd_line_two\(\)](#), and [lcd_write_string_0\(\)](#).

Referenced by [main\(\)](#).

4.8.1.2 DWORD get_fattime (void)

Referenced by [f_open\(\)](#), [f_setlabel\(\)](#), and [f_sync\(\)](#).

4.8.1.3 int main ()

References [order::bank1_note](#), [order::bank1_startstop](#), [order::bank2_note](#), [order::bank2_startstop](#), [order::bank3_note](#), [order::bank3_startstop](#), [onode::data](#), [die\(\)](#), [f_close\(\)](#), [f_getlabel\(\)](#), [f_mount\(\)](#), [f_open\(\)](#), [f_read\(\)](#), [f_write\(\)](#), [FA_CREATE_ALWAYS](#), [FA_READ](#), [FA_WRITE](#), [FR_OK](#), [getOrderNode\(\)](#), [getOrderNote\(\)](#), [getOrderStartstop\(\)](#), [order::id](#), [lcd_clear_and_home\(\)](#), [lcd_init\(\)](#), [lcd_line_one\(\)](#), [lcd_write_string_0\(\)](#), [newNode\(\)](#), [pushNode\(\)](#), [sort\(\)](#), [wavegen_clock_init\(\)](#), [wavegen_noiseWave\(\)](#), [wavegen_pwmInit\(\)](#), [wavegen_setFrequency\(\)](#), [wavegen_setFrequency2\(\)](#), [wavegen_setSound\(\)](#), and [wavegen_sineWave\(\)](#).

4.8.2 Variable Documentation

4.8.2.1 FATFS FatFs

4.8.2.2 FIL* fp

4.9 Makefile File Reference

Variables

- [DEVICE](#)
- gc sections [WI](#)

4.9.1 Variable Documentation

4.9.1.1 [DEVICE](#)

Initial value:

```
=atxmega256d3
AVRDUDE_DEVICE=x256d3

PROG= -c avrispmkII -P usb

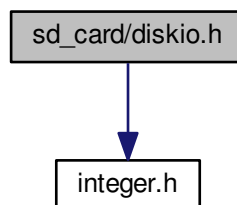
CFLAGS=-g -Wall -mcall-prologues -mmcu=$(DEVICE) -Os -std=c99 -fno-strict-aliasing -
g
CC=avr-gcc
OBJ2HEX=avr-objcopy
LDFLAGS=-Wl
```

4.9.1.2 gc sections [WI](#)

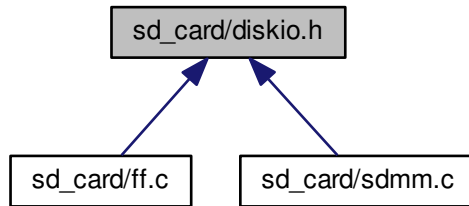
4.10 [sd_card/diskio.h](#) File Reference

```
#include "integer.h"
```

Include dependency graph for [diskio.h](#):



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



Macros

- `#define STA_NOINIT 0x01 /* Drive not initialized */`
- `#define STA_NODISK 0x02 /* No medium in the drive */`
- `#define STA_PROTECT 0x04 /* Write protected */`
- `#define CTRL_SYNC 0 /* Flush disk cache (for write functions) */`
- `#define GET_SECTOR_COUNT 1 /* Get media size (for only f_mkfs()) */`
- `#define GET_BLOCK_SIZE 3 /* Get erase block size (for only f_mkfs()) */`
- `#define CTRL_ERASE_SECTOR 4 /* Force erased a block of sectors (for only _USE_ERASE) */`
- `#define CTRL_POWER 5 /* Get/Set power status */`
- `#define MMC_GET_TYPE 10 /* Get card type */`
- `#define MMC_GET_CSD 11 /* Get CSD */`
- `#define MMC_GET_CID 12 /* Get CID */`
- `#define MMC_GET_OCR 13 /* Get OCR */`
- `#define MMC_GET_SDSTAT 14 /* Get SD status */`
- `#define ATA_GET_REV 20 /* Get F/W revision */`
- `#define ATA_GET_MODEL 21 /* Get model name */`
- `#define ATA_GET_SN 22 /* Get serial number */`
- `#define CT_MMC 0x01 /* MMC ver 3 */`
- `#define CT_SD1 0x02 /* SD ver 1 */`
- `#define CT_SD2 0x04 /* SD ver 2 */`
- `#define CT_SDC (CT_SD1|CT_SD2) /* SD */`
- `#define CT_BLOCK 0x08 /* Block addressing */`

Typedefs

- `typedef BYTE DSTATUS`

Enumerations

- `enum DRESULT {
 RES_OK = 0, RES_ERROR, RES_WRPRT, RES_NOTRDY,
 RES_PARERR }`

Functions

- [DSTATUS disk_initialize](#) (BYTE pdrv)
- [DSTATUS disk_status](#) (BYTE pdrv)
- [DRESULT disk_read](#) (BYTE pdrv, BYTE *buff, DWORD sector, BYTE count)
- [DRESULT disk_write](#) (BYTE pdrv, const BYTE *buff, DWORD sector, BYTE count)
- [DRESULT disk_ioctl](#) (BYTE pdrv, BYTE cmd, void *buff)

4.10.1 Macro Definition Documentation

4.10.1.1 `#define ATA_GET_MODEL 21 /* Get model name */`

4.10.1.2 `#define ATA_GET_REV 20 /* Get F/W revision */`

4.10.1.3 `#define ATA_GET_SN 22 /* Get serial number */`

4.10.1.4 `#define CT_BLOCK 0x08 /* Block addressing */`

Referenced by `disk_initialize()`, `disk_read()`, and `disk_write()`.

4.10.1.5 `#define CT_MMC 0x01 /* MMC ver 3 */`

Referenced by `disk_initialize()`.

4.10.1.6 `#define CT_SD1 0x02 /* SD ver 1 */`

Referenced by `disk_initialize()`.

4.10.1.7 `#define CT_SD2 0x04 /* SD ver 2 */`

Referenced by `disk_initialize()`.

4.10.1.8 `#define CT_SDC (CT_SD1|CT_SD2) /* SD */`

Referenced by `disk_write()`.

4.10.1.9 `#define CTRL_ERASE_SECTOR 4 /* Force erased a block of sectors (for only _USE_ERASE) */`

Referenced by `remove_chain()`.

4.10.1.10 `#define CTRL_POWER 5 /* Get/Set power status */`

4.10.1.11 `#define CTRL_SYNC 0 /* Flush disk cache (for write functions) */`

Referenced by `disk_ioctl()`, and `sync_fs()`.

4.10.1.12 **#define GET_BLOCK_SIZE 3** /* Get erase block size (for only f_mkfs()) */

Referenced by disk_ioctl().

4.10.1.13 **#define GET_SECTOR_COUNT 1** /* Get media size (for only f_mkfs()) */

Referenced by disk_ioctl().

4.10.1.14 **#define MMC_GET_CID 12** /* Get CID */

4.10.1.15 **#define MMC_GET_CSD 11** /* Get CSD */

4.10.1.16 **#define MMC_GET_OCR 13** /* Get OCR */

4.10.1.17 **#define MMC_GET_SDSTAT 14** /* Get SD status */

4.10.1.18 **#define MMC_GET_TYPE 10** /* Get card type */

4.10.1.19 **#define STA_NODISK 0x02** /* No medium in the drive */

4.10.1.20 **#define STA_NOINIT 0x01** /* Drive not initialized */

Referenced by chk_mounted(), disk_initialize(), disk_ioctl(), disk_read(), disk_status(), disk_write(), and validate().

4.10.1.21 **#define STA_PROTECT 0x04** /* Write protected */

Referenced by chk_mounted().

4.10.2 Typedef Documentation

4.10.2.1 **typedef BYTE DSTATUS**

4.10.3 Enumeration Type Documentation

4.10.3.1 **enum DRESULT**

Enumerator

RES_OK

RES_ERROR

RES_WRPRT

RES_NOTRDY

RES_PARERR

4.10.4 Function Documentation

4.10.4.1 DSTATUS disk_initialize (BYTE pdrv)

References ACMD41, CardType, CK_INIT, CK_L, CMD0, CMD1, CMD16, CMD58, CMD8, CS_H, CS_INIT, CT_BLOCK, CT_MMC, CT_SD1, CT_SD2, deselect(), DI_INIT, dly_us(), DO_INIT, rcvr_mmc(), RES_NOTRDY, send_cmd(), STA_NOINIT, and Stat.

Referenced by chk_mounted().

4.10.4.2 DRESULT disk_ioctl (BYTE pdrv, BYTE cmd, void * buff)

References CMD9, CTRL_SYNC, deselect(), disk_status(), GET_BLOCK_SIZE, GET_SECTOR_COUNT, rcvr_datablock(), RES_ERROR, RES_NOTRDY, RES_OK, RES_PARERR, select(), send_cmd(), and STA_NOINIT.

Referenced by chk_mounted(), remove_chain(), and sync_fs().

4.10.4.3 DRESULT disk_read (BYTE pdrv, BYTE * buff, DWORD sector, BYTE count)

References CardType, CMD12, CMD17, CMD18, CT_BLOCK, deselect(), disk_status(), rcvr_datablock(), RES_ERROR, RES_NOTRDY, RES_OK, send_cmd(), and STA_NOINIT.

Referenced by check_fs(), chk_mounted(), f_lseek(), f_read(), f_write(), and move_window().

4.10.4.4 DSTATUS disk_status (BYTE pdrv)

References STA_NOINIT, and Stat.

Referenced by chk_mounted(), disk_ioctl(), disk_read(), disk_write(), and validate().

4.10.4.5 DRESULT disk_write (BYTE pdrv, const BYTE * buff, DWORD sector, BYTE count)

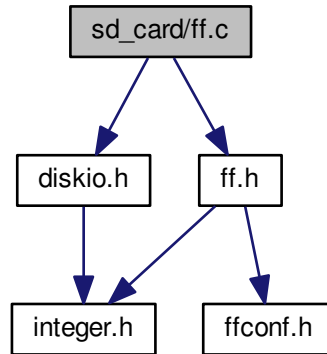
References ACMD23, CardType, CMD24, CMD25, CT_BLOCK, CT_SDC, deselect(), disk_status(), RES_ERROR, RES_NOTRDY, RES_OK, send_cmd(), STA_NOINIT, and xmit_datablock().

Referenced by f_lseek(), f_read(), f_sync(), f_write(), sync_fs(), and sync_window().

4.11 sd_card/ff.c File Reference

```
#include "ff.h"
#include "diskio.h"
```

Include dependency graph for ff.c:



Macros

- `#define SS(fs) 512U /* Fixed sector size */`
- `#define ENTER_FF(fs)`
- `#define LEAVE_FF(fs, res) return res`
- `#define ABORT(fs, res) { fp->flag |= FA__ERROR; LEAVE_FF(fs, res); }`
- `#define _DF1S 0`
- `#define _EXCVT`
- `#define IsUpper(c) (((c)>='A')&&((c)<='Z'))`
- `#define IsLower(c) (((c)>='a')&&((c)<='z'))`
- `#define IsDigit(c) (((c)>='0')&&((c)<='9'))`
- `#define IsDBCS1(c) 0`
- `#define IsDBCS2(c) 0`
- `#define NS 11 /* Index of name status byte in fn[] */`
- `#define NS_LOSS 0x01 /* Out of 8.3 format */`
- `#define NS_LFN 0x02 /* Force to create LFN entry */`
- `#define NS_LAST 0x04 /* Last segment */`
- `#define NS_BODY 0x08 /* Lower case flag (body) */`
- `#define NS_EXT 0x10 /* Lower case flag (ext) */`
- `#define NS_DOT 0x20 /* Dot entry */`
- `#define MIN_FAT16 4086 /* Minimum number of clusters for FAT16 */`
- `#define MIN_FAT32 65526 /* Minimum number of clusters for FAT32 */`
- `#define BS_jmpBoot 0 /* Jump instruction (3) */`
- `#define BS_OEMName 3 /* OEM name (8) */`
- `#define BPB_BytsPerSec 11 /* Sector size [byte] (2) */`
- `#define BPB_SecPerClus 13 /* Cluster size [sector] (1) */`
- `#define BPB_RsvdSecCnt 14 /* Size of reserved area [sector] (2) */`
- `#define BPB_NumFATs 16 /* Number of FAT copies (1) */`
- `#define BPB_RootEntCnt 17 /* Number of root dir entries for FAT12/16 (2) */`
- `#define BPB_TotSec16 19 /* Volume size [sector] (2) */`

- #define `BPB_Media` 21 /* Media descriptor (1) */
- #define `BPB_FATSz16` 22 /* FAT size [sector] (2) */
- #define `BPB_SecPerTrk` 24 /* Track size [sector] (2) */
- #define `BPB_NumHeads` 26 /* Number of heads (2) */
- #define `BPB_HiddSec` 28 /* Number of special hidden sectors (4) */
- #define `BPB_TotSec32` 32 /* Volume size [sector] (4) */
- #define `BS_DrvNum` 36 /* Physical drive number (2) */
- #define `BS_BootSig` 38 /* Extended boot signature (1) */
- #define `BS_VolID` 39 /* Volume serial number (4) */
- #define `BS_VolLab` 43 /* Volume label (8) */
- #define `BS_FilSysType` 54 /* File system type (1) */
- #define `BPB_FATSz32` 36 /* FAT size [sector] (4) */
- #define `BPB_ExtFlags` 40 /* Extended flags (2) */
- #define `BPB_FSVer` 42 /* File system version (2) */
- #define `BPB_RootClus` 44 /* Root dir first cluster (4) */
- #define `BPB_FSInfo` 48 /* Offset of FSInfo sector (2) */
- #define `BPB_BkBootSec` 50 /* Offset of backup boot sector (2) */
- #define `BS_DrvNum32` 64 /* Physical drive number (2) */
- #define `BS_BootSig32` 66 /* Extended boot signature (1) */
- #define `BS_VolID32` 67 /* Volume serial number (4) */
- #define `BS_VolLab32` 71 /* Volume label (8) */
- #define `BS_FilSysType32` 82 /* File system type (1) */
- #define `FSI_LeadSig` 0 /* FSI: Leading signature (4) */
- #define `FSI_StrucSig` 484 /* FSI: Structure signature (4) */
- #define `FSI_Free_Count` 488 /* FSI: Number of free clusters (4) */
- #define `FSI_Nxt_Free` 492 /* FSI: Last allocated cluster (4) */
- #define `MBR_Table` 446 /* MBR: Partition table offset (2) */
- #define `SZ_PTE` 16 /* MBR: Size of a partition table entry */
- #define `BS_55AA` 510 /* Boot sector signature (2) */
- #define `DIR_Name` 0 /* Short file name (11) */
- #define `DIR_Attr` 11 /* Attribute (1) */
- #define `DIR_NTres` 12 /* NT flag (1) */
- #define `DIR_CrtTimeTenth` 13 /* Created time sub-second (1) */
- #define `DIR_CrtTime` 14 /* Created time (2) */
- #define `DIR_CrtDate` 16 /* Created date (2) */
- #define `DIR_LstAccDate` 18 /* Last accessed date (2) */
- #define `DIR_FstClusHI` 20 /* Higher 16-bit of first cluster (2) */
- #define `DIR_WrtTime` 22 /* Modified time (2) */
- #define `DIR_WrtDate` 24 /* Modified date (2) */
- #define `DIR_FstClusLO` 26 /* Lower 16-bit of first cluster (2) */
- #define `DIR_FileSize` 28 /* File size (4) */
- #define `LDIR_Ord` 0 /* LFN entry `order` and `LLE` flag (1) */
- #define `LDIR_Attr` 11 /* LFN attribute (1) */
- #define `LDIR_Type` 12 /* LFN type (1) */
- #define `LDIR_Chksum` 13 /* Sum of corresponding SFN entry */
- #define `LDIR_FstClusLO` 26 /* Filled by zero (0) */
- #define `SZ_DIR` 32 /* Size of a directory entry */
- #define `LLE` 0x40 /* Last long entry flag in `LDIR_Ord` */
- #define `DDE` 0xE5 /* Deleted directory entry mark in `DIR_Name[0]` */
- #define `NDDE` 0x05 /* Replacement of the character collides with `DDE` */
- #define `DEF_NAMEBUF` `BYTE` `sfn[12]`
- #define `INIT_BUF`(`dobj`) (`dobj`).fn = `sfn`
- #define `FREE_BUF`()

Functions

- static void `mem_cpy` (void *dst, const void *src, `UINT` cnt)
- static void `mem_set` (void *dst, int val, `UINT` cnt)
- static int `mem_cmp` (const void *dst, const void *src, `UINT` cnt)
- static int `chk_chr` (const char *str, int chr)
- static `FRESULT` `sync_window` (`FATFS` *fs)
- static `FRESULT` `move_window` (`FATFS` *fs, `DWORD` sector)
- static `FRESULT` `sync_fs` (`FATFS` *fs)
- `DWORD` `clust2sect` (`FATFS` *fs, `DWORD` clst)
- `DWORD` `get_fat` (`FATFS` *fs, `DWORD` clst)
- `FRESULT` `put_fat` (`FATFS` *fs, `DWORD` clst, `DWORD` val)
- static `FRESULT` `remove_chain` (`FATFS` *fs, `DWORD` clst)
- static `DWORD` `create_chain` (`FATFS` *fs, `DWORD` clst)
- static `FRESULT` `dir_sdi` (`DIR` *dj, `WORD` idx)
- static `FRESULT` `dir_next` (`DIR` *dj, int stretch)
- static `FRESULT` `dir_alloc` (`DIR` *dj, `UINT` nent)
- static `DWORD` `ld_clust` (`FATFS` *fs, `BYTE` *dir)
- static void `st_clust` (`BYTE` *dir, `DWORD` cl)
- static `FRESULT` `dir_find` (`DIR` *dj)
- static `FRESULT` `dir_read` (`DIR` *dj, int vol)
- static `FRESULT` `dir_register` (`DIR` *dj)
- static `FRESULT` `create_name` (`DIR` *dj, const `TCHAR` **path)
- static `FRESULT` `follow_path` (`DIR` *dj, const `TCHAR` *path)
- static `BYTE` `check_fs` (`FATFS` *fs, `DWORD` sect)
- static `FRESULT` `chk_mounted` (const `TCHAR` **path, `FATFS` **rfs, `BYTE` wmode)
- static `FRESULT` `validate` (void *obj)
- `FRESULT` `f_mount` (`BYTE` vol, `FATFS` *fs)
- `FRESULT` `f_open` (`FIL` *fp, const `TCHAR` *path, `BYTE` mode)
- `FRESULT` `f_read` (`FIL` *fp, void *buff, `UINT` btr, `UINT` *br)
- `FRESULT` `f_write` (`FIL` *fp, const void *buff, `UINT` btw, `UINT` *bw)
- `FRESULT` `f_sync` (`FIL` *fp)
- `FRESULT` `f_close` (`FIL` *fp)
- `FRESULT` `f_lseek` (`FIL` *fp, `DWORD` ofs)
- `FRESULT` `f_getlabel` (const `TCHAR` *path, `TCHAR` *label, `DWORD` *sn)
- `FRESULT` `f_setlabel` (const `TCHAR` *label)

Variables

- static `FATFS` * `FatFs` [`_VOLUMES`]
- static `WORD` `Fsid`
- static const `BYTE` `ExCvt` [] = `_EXCVT`

4.11.1 Macro Definition Documentation

4.11.1.1 `#define _DF1S 0`

Referenced by `create_name()`, and `f_setlabel()`.

4.11.1.2 `#define _EXCVT`**Value:**

```
{ 0x80, 0x9A, 0x90, 0x41, 0x8E, 0x41, 0x8F, 0x80, 0x45, 0x45, 0x45, 0x49, 0x49, 0x49, 0x8E, 0x8F, 0x90, 0x92, 0x4F, 0x99,
  0x4F, 0x55, 0x55, 0x59, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F, \
  0x41, 0x49, 0x4F, 0x55, 0xA5, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0x21, 0xAE,
  0xAF, 0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF, \
  0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE,
  0xCF, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF, \
  0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE,
  0xEF, 0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF }
```

4.11.1.3 `#define ABORT(fs, res) { fp->flag |= FA__ERROR; LEAVE_FF(fs, res); }`

Referenced by `f_lseek()`, `f_read()`, and `f_write()`.

4.11.1.4 `#define BPB_BkBootSec 50 /* Offset of backup boot sector (2) */`4.11.1.5 `#define BPB_BytsPerSec 11 /* Sector size [byte] (2) */`

Referenced by `chk_mounted()`.

4.11.1.6 `#define BPB_ExtFlags 40 /* Extended flags (2) */`4.11.1.7 `#define BPB_FATSz16 22 /* FAT size [sector] (2) */`

Referenced by `chk_mounted()`.

4.11.1.8 `#define BPB_FATSz32 36 /* FAT size [sector] (4) */`

Referenced by `chk_mounted()`.

4.11.1.9 `#define BPB_FSInfo 48 /* Offset of FSInfo sector (2) */`

Referenced by `chk_mounted()`.

4.11.1.10 `#define BPB_FSVer 42 /* File system version (2) */`4.11.1.11 `#define BPB_HiddSec 28 /* Number of special hidden sectors (4) */`4.11.1.12 `#define BPB_Media 21 /* Media descriptor (1) */`4.11.1.13 `#define BPB_NumFATs 16 /* Number of FAT copies (1) */`

Referenced by `chk_mounted()`.

4.11.1.14 **#define BPB_NumHeads 26** /* Number of heads (2) */

4.11.1.15 **#define BPB_RootClus 44** /* Root dir first cluster (4) */

Referenced by `chk_mounted()`.

4.11.1.16 **#define BPB_RootEntCnt 17** /* Number of root dir entries for FAT12/16 (2) */

Referenced by `chk_mounted()`.

4.11.1.17 **#define BPB_RsvdSecCnt 14** /* Size of reserved area [sector] (2) */

Referenced by `chk_mounted()`.

4.11.1.18 **#define BPB_SecPerClus 13** /* Cluster size [sector] (1) */

Referenced by `chk_mounted()`.

4.11.1.19 **#define BPB_SecPerTrk 24** /* Track size [sector] (2) */

4.11.1.20 **#define BPB_TotSec16 19** /* Volume size [sector] (2) */

Referenced by `chk_mounted()`.

4.11.1.21 **#define BPB_TotSec32 32** /* Volume size [sector] (4) */

Referenced by `chk_mounted()`.

4.11.1.22 **#define BS_55AA 510** /* Boot sector signature (2) */

Referenced by `check_fs()`, `chk_mounted()`, and `sync_fs()`.

4.11.1.23 **#define BS_BootSig 38** /* Extended boot signature (1) */

4.11.1.24 **#define BS_BootSig32 66** /* Extended boot signature (1) */

4.11.1.25 **#define BS_DrvNum 36** /* Physical drive number (2) */

4.11.1.26 **#define BS_DrvNum32 64** /* Physical drive number (2) */

4.11.1.27 **#define BS_FilSysType 54** /* File system type (1) */

Referenced by `check_fs()`.

4.11.1.28 **#define BS_FilSysType32 82** /* File system type (1) */

Referenced by `check_fs()`.

4.11.1.29 `#define BS_impBoot 0 /* Jump instruction (3) */`

4.11.1.30 `#define BS_OEMName 3 /* OEM name (8) */`

4.11.1.31 `#define BS_VolID 39 /* Volume serial number (4) */`

Referenced by `f_getlabel()`.

4.11.1.32 `#define BS_VolID32 67 /* Volume serial number (4) */`

Referenced by `f_getlabel()`.

4.11.1.33 `#define BS_VolLab 43 /* Volume label (8) */`

4.11.1.34 `#define BS_VolLab32 71 /* Volume label (8) */`

4.11.1.35 `#define DDE 0xE5 /* Deleted directory entry mark in DIR_Name[0] */`

Referenced by `create_name()`, `dir_alloc()`, `dir_find()`, `dir_read()`, and `f_setlabel()`.

4.11.1.36 `#define DEF_NAMEBUF BYTE sfn[12]`

Referenced by `f_open()`.

4.11.1.37 `#define DIR_Attr 11 /* Attribute (1) */`

Referenced by `dir_find()`, `dir_read()`, `f_open()`, `f_setlabel()`, `f_sync()`, and `follow_path()`.

4.11.1.38 `#define DIR_CrtDate 16 /* Created date (2) */`

4.11.1.39 `#define DIR_CrtTime 14 /* Created time (2) */`

Referenced by `f_open()`.

4.11.1.40 `#define DIR_CrtTimeTenth 13 /* Created time sub-second (1) */`

4.11.1.41 `#define DIR_FileSize 28 /* File size (4) */`

Referenced by `f_open()`, and `f_sync()`.

4.11.1.42 `#define DIR_FstClusHI 20 /* Higher 16-bit of first cluster (2) */`

Referenced by `ld_clust()`, and `st_clust()`.

4.11.1.43 `#define DIR_FstClusLO 26 /* Lower 16-bit of first cluster (2) */`

Referenced by `ld_clust()`, and `st_clust()`.

4.11.1.44 **#define DIR_LstAccDate 18** /* Last accessed date (2) */

Referenced by `f_sync()`.

4.11.1.45 **#define DIR_Name 0** /* Short file name (11) */

Referenced by `dir_find()`, and `dir_read()`.

4.11.1.46 **#define DIR_NTres 12** /* NT flag (1) */

Referenced by `dir_register()`.

4.11.1.47 **#define DIR_WrtDate 24** /* Modified date (2) */

4.11.1.48 **#define DIR_WrtTime 22** /* Modified time (2) */

Referenced by `f_setlabel()`, and `f_sync()`.

4.11.1.49 **#define ENTER_FF(fs)**

Referenced by `chk_mounted()`, and `validate()`.

4.11.1.50 **#define FREE_BUF()**

Referenced by `f_open()`.

4.11.1.51 **#define FSI_Free_Count 488** /* FSI: Number of free clusters (4) */

Referenced by `chk_mounted()`, and `sync_fs()`.

4.11.1.52 **#define FSI_LeadSig 0** /* FSI: Leading signature (4) */

Referenced by `chk_mounted()`, and `sync_fs()`.

4.11.1.53 **#define FSI_Nxt_Free 492** /* FSI: Last allocated cluster (4) */

Referenced by `chk_mounted()`, and `sync_fs()`.

4.11.1.54 **#define FSI_StrucSig 484** /* FSI: Structure signature (4) */

Referenced by `chk_mounted()`, and `sync_fs()`.

4.11.1.55 **#define INIT_BUF(dobj)**(dobj).fn = sfn

Referenced by `f_open()`.

4.11.1.56 `#define IsDBCS1(c) 0`

Referenced by `create_name()`, `f_getlabel()`, and `f_setlabel()`.

4.11.1.57 `#define IsDBCS2(c) 0`

Referenced by `create_name()`, `f_getlabel()`, and `f_setlabel()`.

4.11.1.58 `#define IsDigit(c) (((c)>='0')&&((c)<='9'))`

4.11.1.59 `#define IsLower(c) (((c)>='a')&&((c)<='z'))`

Referenced by `create_name()`, and `f_setlabel()`.

4.11.1.60 `#define IsUpper(c) (((c)>='A')&&((c)<='Z'))`

Referenced by `create_name()`.

4.11.1.61 `#define LDIR_Attr 11 /* LFN attribute (1) */`

4.11.1.62 `#define LDIR_Chksum 13 /* Sum of corresponding SFN entry */`

Referenced by `dir_find()`, and `dir_read()`.

4.11.1.63 `#define LDIR_FstClusLO 26 /* Filled by zero (0) */`

4.11.1.64 `#define LDIR_Ord 0 /* LFN entry order and LLE flag (1) */`

4.11.1.65 `#define LDIR_Type 12 /* LFN type (1) */`

4.11.1.66 `#define LEAVE_FF(fs, res) return res`

Referenced by `f_close()`, `f_getlabel()`, `f_lseek()`, `f_open()`, `f_read()`, `f_setlabel()`, `f_sync()`, and `f_write()`.

4.11.1.67 `#define LLE 0x40 /* Last long entry flag in LDIR_Ord */`

4.11.1.68 `#define MBR_Table 446 /* MBR: Partition table offset (2) */`

Referenced by `chk_mounted()`.

4.11.1.69 `#define MIN_FAT16 4086 /* Minimum number of clusters for FAT16 */`

Referenced by `chk_mounted()`.

4.11.1.70 `#define MIN_FAT32 65526 /* Minimum number of clusters for FAT32 */`

Referenced by `chk_mounted()`.

4.11.1.71 **#define NDDE 0x05** /* Replacement of the character collides with DDE */

Referenced by `create_name()`.

4.11.1.72 **#define NS 11** /* Index of name status byte in `fn[]` */

Referenced by `create_name()`, `dir_find()`, `dir_register()`, and `follow_path()`.

4.11.1.73 **#define NS_BODY 0x08** /* Lower case flag (body) */

Referenced by `create_name()`, and `dir_register()`.

4.11.1.74 **#define NS_DOT 0x20** /* Dot entry */

Referenced by `create_name()`, `dir_register()`, and `follow_path()`.

4.11.1.75 **#define NS_EXT 0x10** /* Lower case flag (ext) */

Referenced by `create_name()`, and `dir_register()`.

4.11.1.76 **#define NS_LAST 0x04** /* Last segment */

Referenced by `create_name()`, and `follow_path()`.

4.11.1.77 **#define NS_LFN 0x02** /* Force to create LFN entry */

Referenced by `create_name()`, and `dir_register()`.

4.11.1.78 **#define NS_LOSS 0x01** /* Out of 8.3 format */

Referenced by `create_name()`, `dir_find()`, and `dir_register()`.

4.11.1.79 **#define SS(fs) 512U** /* Fixed sector size */

Referenced by `chk_mounted()`, `dir_next()`, `dir_sdi()`, `f_lseek()`, `f_read()`, `f_write()`, `get_fat()`, and `put_fat()`.

4.11.1.80 **#define SZ_DIR 32** /* Size of a directory entry */

Referenced by `chk_mounted()`, `dir_next()`, `dir_register()`, `dir_sdi()`, and `f_setlabel()`.

4.11.1.81 **#define SZ_PTE 16** /* MBR: Size of a partition table entry */

Referenced by `chk_mounted()`.

4.11.2 Function Documentation

4.11.2.1 static BYTE check_fs (FATFS * *fs*, DWORD *sect*) [static]

References BS_55AA, BS_FilSysType, BS_FilSysType32, disk_read(), FATFS::drv, LD_DWORD, LD_WORD, RES_OK, and FATFS::win.

Referenced by chk_mounted().

4.11.2.2 static int chk_chr (const char * *str*, int *chr*) [static]

Referenced by create_name(), and f_setlabel().

4.11.2.3 static FRESULT chk_mounted (const TCHAR ** *path*, FATFS ** *rfs*, BYTE *wmode*) [static]

References _FS_READONLY, _VOLUMES, BPB_BytsPerSec, BPB_FATSz16, BPB_FATSz32, BPB_FSInfo, BPB_NumFATs, BPB_RootClus, BPB_RootEntCnt, BPB_RsvdSecCnt, BPB_SecPerClus, BPB_TotSec16, BPB_TotSec32, BS_55AA, check_fs(), FATFS::csize, FATFS::database, FATFS::dirbase, disk_initialize(), disk_ioctl(), disk_read(), disk_status(), FATFS::drv, ENTER_FF, FATFS::fatbase, FR_DISK_ERR, FR_INVALID_DRIVE, FR_NO_FILESYSTEM, FR_NOT_ENABLED, FR_NOT_READY, FR_OK, FR_WRITE_PROTECTED, FATFS::free_clust, FS_FAT12, FS_FAT16, FS_FAT32, FATFS::fs_type, FATFS::fsi_flag, FSI_Free_Count, FSI_LeadSig, FSI_Nxt_Free, FATFS::fsi_sector, FSI_StrucSig, Fsid, FATFS::fsize, FATFS::id, FATFS::last_clust, LD2PD, LD2PT, LD_DWORD, LD_WORD, MBR_Table, MIN_FAT16, MIN_FAT32, FATFS::n_fatent, FATFS::n_fats, FATFS::n_rootdir, RES_OK, SS, STA_NOINIT, STA_PROTECT, SZ_DIR, SZ_PTE, FATFS::volbase, FATFS::wflag, FATFS::win, and FATFS::winsect.

Referenced by f_getlabel(), f_open(), and f_setlabel().

4.11.2.4 DWORD clust2sect (FATFS * *fs*, DWORD *clst*)

References FATFS::csize, FATFS::database, and FATFS::n_fatent.

Referenced by dir_next(), dir_sdi(), f_lseek(), f_read(), f_write(), and remove_chain().

4.11.2.5 static DWORD create_chain (FATFS * *fs*, DWORD *clst*) [static]

References FR_DISK_ERR, FR_OK, FATFS::free_clust, FATFS::fsi_flag, get_fat(), FATFS::last_clust, FATFS::n_fatent, and put_fat().

Referenced by dir_next(), f_lseek(), and f_write().

4.11.2.6 static FRESULT create_name (DIR * *dj*, const TCHAR ** *path*) [static]

References _DF1S, _MAX_LFN, chk_chr(), DDE, ExCvt, DIR::fn, FR_INVALID_NAME, FR_OK, IsDBCS1, IsDBCS2, IsLower, IsUpper, mem_set(), NDDE, NS, NS_BODY, NS_DOT, NS_EXT, NS_LAST, NS_LFN, and NS_LOSS.

Referenced by follow_path().

4.11.2.7 static FRESULT dir_alloc (DIR * *dj*, UINT *nent*) [static]

References DDE, DIR::dir, dir_next(), dir_sdi(), FR_OK, DIR::fs, move_window(), and DIR::sect.

Referenced by dir_register(), and f_setlabel().

4.11.2.8 static FRESULT dir_find (DIR * *dj*) [static]

References AM_LFN, AM_MASK, AM_VOL, DDE, DIR::dir, DIR_Attr, DIR_Name, dir_next(), dir_sdi(), DIR::fn, FR_NO_FILE, FR_OK, DIR::fs, DIR::index, LDIR_Chksum, mem_cmp(), move_window(), NS, NS_LOSS, and DIR::sect.

Referenced by dir_register(), and follow_path().

4.11.2.9 static FRESULT dir_next (DIR * *dj*, int *stretch*) [static]

References DIR::clust, clust2sect(), create_chain(), FATFS::csize, DIR::dir, FR_DENIED, FR_DISK_ERR, FR_INT_ERR, FR_NO_FILE, FR_OK, DIR::fs, get_fat(), DIR::index, mem_set(), FATFS::n_fatent, FATFS::n_rootdir, DIR::sect, SS, sync_window(), SZ_DIR, FATFS::wflag, FATFS::win, and FATFS::winsect.

Referenced by dir_alloc(), dir_find(), dir_read(), and dir_register().

4.11.2.10 static FRESULT dir_read (DIR * *dj*, int *vol*) [static]

References _FS_RPATH, AM_LFN, AM_MASK, AM_VOL, DDE, DIR::dir, DIR_Attr, DIR_Name, dir_next(), FR_NO_FILE, FR_OK, DIR::fs, DIR::index, LDIR_Chksum, move_window(), and DIR::sect.

Referenced by f_getlabel(), and f_setlabel().

4.11.2.11 static FRESULT dir_register (DIR * *dj*) [static]

References _FS_RPATH, DIR::dir, dir_alloc(), dir_find(), dir_next(), DIR_NTres, dir_sdi(), DIR::fn, FR_DENIED, FR_INVALID_NAME, FR_NO_FILE, FR_OK, DIR::fs, DIR::index, mem_cpy(), mem_set(), move_window(), NS, NS_BODY, NS_DOT, NS_EXT, NS_LFN, NS_LOSS, DIR::sect, SZ_DIR, and FATFS::wflag.

Referenced by f_open().

4.11.2.12 static FRESULT dir_sdi (DIR * *dj*, WORD *idx*) [static]

References DIR::clust, clust2sect(), FATFS::csize, DIR::dir, FATFS::dirbase, FR_DISK_ERR, FR_INT_ERR, FR_OK, DIR::fs, FS_FAT32, FATFS::fs_type, get_fat(), DIR::index, FATFS::n_fatent, FATFS::n_rootdir, DIR::sclust, DIR::sect, SS, SZ_DIR, and FATFS::win.

Referenced by dir_alloc(), dir_find(), dir_register(), f_getlabel(), f_setlabel(), and follow_path().

4.11.2.13 FRESULT f_close (FIL * *fp*)

References f_sync(), FR_OK, FIL::fs, LEAVE_FF, and validate().

Referenced by main().

4.11.2.14 FRESULT f_getlabel (const TCHAR * *path*, TCHAR * *label*, DWORD * *sn*)

References BS_VolID, BS_VolID32, chk_mounted(), DIR::dir, dir_read(), dir_sdi(), FR_NO_FILE, FR_OK, DIR::fs, FS_FAT32, FATFS::fs_type, IsDBCS1, IsDBCS2, LD_DWORD, LEAVE_FF, mem_cpy(), move_window(), DIR::sclust, FATFS::volbase, and FATFS::win.

Referenced by main().

4.11.2.15 FRESULT f_lseek (FIL * *fp*, DWORD *ofs*)

References _FS_READONLY, ABORT, FIL::clust, clust2sect(), create_chain(), CREATE_LINKMAP, FATFS::csize, disk_read(), disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__ERROR, FA__WRITTEN, FA_WRITE, FIL::flag, FIL::fptr, FR_DISK_ERR, FR_INT_ERR, FR_NOT_ENOUGH_CORE, FR_OK, FIL::fs, FIL::fsize, get_fat(), LEAVE_FF, FATFS::n_fatent, RES_OK, FIL::sclust, SS, and validate().

4.11.2.16 FRESULT f_mount (BYTE *vol*, FATFS * *fs*)

References _VOLUMES, FR_INT_ERR, FR_INVALID_DRIVE, FR_OK, and FATFS::fs_type.

Referenced by main().

4.11.2.17 FRESULT f_open (FIL * *fp*, const TCHAR * *path*, BYTE *mode*)

References AM_DIR, AM_RDO, chk_mounted(), DEF_NAMEBUF, DIR::dir, DIR_Attr, DIR_CrtTime, DIR_FileSize, FIL::dir_ptr, dir_register(), FIL::dir_sect, FIL::dsect, FA__WRITTEN, FA_CREATE_ALWAYS, FA_CREATE_NEW, FA_OPEN_ALWAYS, FA_READ, FA_WRITE, FIL::flag, follow_path(), FIL::fptr, FR_DENIED, FR_EXIST, FR_INT_ERR, FR_INVALID_NAME, FR_INVALID_OBJECT, FR_NO_FILE, FR_OK, FR_TOO_MANY_OPEN_FILES, FREE_BUF, FIL::fs, DIR::fs, FIL::fsize, get_fattime(), FATFS::id, FIL::id, INIT_BUF, FATFS::last_clust, ld_clust(), LD_DWORD, LEAVE_FF, move_window(), remove_chain(), FIL::sclust, st_clust(), ST_DWORD, FATFS::wflag, and FATFS::winsect.

Referenced by main().

4.11.2.18 FRESULT f_read (FIL * *fp*, void * *buff*, UINT *btr*, UINT * *br*)

References ABORT, FIL::clust, clust2sect(), FATFS::csize, disk_read(), disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__ERROR, FA_READ, FIL::flag, FIL::fptr, FR_DENIED, FR_DISK_ERR, FR_INT_ERR, FR_OK, FIL::fs, FIL::fsize, get_fat(), LEAVE_FF, mem_cpy(), move_window(), RES_OK, FIL::sclust, SS, validate(), FATFS::wflag, FATFS::win, and FATFS::winsect.

Referenced by main().

4.11.2.19 FRESULT f_setlabel (const TCHAR * *label*)

References _DF1S, AM_VOL, chk_chr(), chk_mounted(), DDE, DIR::dir, dir_alloc(), DIR_Attr, dir_read(), dir_sdi(), DIR_WrtTime, ExCvt, FR_INVALID_NAME, FR_NO_FILE, FR_OK, DIR::fs, get_fattime(), IsDBCS1, IsDBCS2, IsLower, LEAVE_FF, mem_cpy(), mem_set(), DIR::sclust, ST_DWORD, sync_fs(), SZ_DIR, and FATFS::wflag.

4.11.2.20 FRESULT f_sync (FIL * *fp*)

References AM_ARC, DIR_Attr, DIR_FileSize, DIR_LstAccDate, FIL::dir_ptr, FIL::dir_sect, DIR_WrtTime, disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__WRITTEN, FIL::flag, FR_DISK_ERR, FR_OK, FIL::fs, FIL::fsize, get_fattime(), LEAVE_FF, move_window(), RES_OK, FIL::sclust, st_clust(), ST_DWORD, ST_WORD, sync_fs(), validate(), and FATFS::wflag.

Referenced by f_close().

4.11.2.21 FRESULT f_write (FIL * *fp*, const void * *buff*, UINT *btw*, UINT * *bw*)

References ABORT, FIL::clust, clust2sect(), create_chain(), FATFS::csize, disk_read(), disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__ERROR, FA__WRITTEN, FA_WRITE, FIL::flag, FIL::fptr, FR_DENIED, FR_DISK_ERR,

FR_INT_ERR, FR_OK, FIL::fs, FIL::fsize, LEAVE_FF, mem_cpy(), move_window(), RES_OK, FIL::sclust, SS, sync_window(), validate(), FATFS::wflag, FATFS::win, and FATFS::winsect.

Referenced by main().

4.11.2.22 static FRESULT follow_path (DIR * *dj*, const TCHAR * *path*) [static]

References _FS_RPATH, AM_DIR, create_name(), DIR::dir, DIR_Attr, dir_find(), dir_sdi(), DIR::fn, FR_NO_FILE, FR_NO_PATH, FR_OK, DIR::fs, ld_clust(), NS, NS_DOT, NS_LAST, and DIR::sclust.

Referenced by f_open().

4.11.2.23 DWORD get_fat (FATFS * *fs*, DWORD *clst*)

References FATFS::fatbase, FS_FAT12, FS_FAT16, FS_FAT32, FATFS::fs_type, LD_DWORD, LD_WORD, move_window(), FATFS::n_fatent, SS, and FATFS::win.

Referenced by create_chain(), dir_next(), dir_sdi(), f_lseek(), f_read(), and remove_chain().

4.11.2.24 static DWORD ld_clust (FATFS * *fs*, BYTE * *dir*) [static]

References DIR_FstClusHI, DIR_FstClusLO, FS_FAT32, FATFS::fs_type, and LD_WORD.

Referenced by f_open(), and follow_path().

4.11.2.25 static int mem_cmp (const void * *dst*, const void * *src*, UINT *cnt*) [static]

Referenced by dir_find().

4.11.2.26 static void mem_cpy (void * *dst*, const void * *src*, UINT *cnt*) [static]

Referenced by dir_register(), f_getlabel(), f_read(), f_setlabel(), and f_write().

4.11.2.27 static void mem_set (void * *dst*, int *val*, UINT *cnt*) [static]

Referenced by create_name(), dir_next(), dir_register(), f_setlabel(), and sync_fs().

4.11.2.28 static FRESULT move_window (FATFS * *fs*, DWORD *sector*) [static]

References disk_read(), FATFS::drv, FR_DISK_ERR, FR_OK, RES_OK, sync_window(), FATFS::win, and FATFS::winsect.

Referenced by dir_alloc(), dir_find(), dir_read(), dir_register(), f_getlabel(), f_open(), f_read(), f_sync(), f_write(), get_fat(), and put_fat().

4.11.2.29 FRESULT put_fat (FATFS * *fs*, DWORD *clst*, DWORD *val*)

References FATFS::fatbase, FR_INT_ERR, FR_OK, FS_FAT12, FS_FAT16, FS_FAT32, FATFS::fs_type, LD_DWORD, move_window(), FATFS::n_fatent, SS, ST_DWORD, ST_WORD, FATFS::wflag, and FATFS::win.

Referenced by create_chain(), and remove_chain().

4.11.2.30 static FRESULT remove_chain (FATFS * *fs*, DWORD *clst*) [static]

References clust2sect(), FATFS::csize, CTRL_ERASE_SECTOR, disk_ioctl(), FATFS::drv, FR_DISK_ERR, FR_INT_ERR, FR_OK, FATFS::free_clust, FATFS::fsi_flag, get_fat(), FATFS::n_fatent, and put_fat().

Referenced by f_open().

4.11.2.31 static void st_clust (BYTE * *dir*, DWORD *cl*) [static]

References DIR_FstClusHI, DIR_FstClusLO, and ST_WORD.

Referenced by f_open(), and f_sync().

4.11.2.32 static FRESULT sync_fs (FATFS * *fs*) [static]

References BS_55AA, CTRL_SYNC, disk_ioctl(), disk_write(), FATFS::drv, FR_DISK_ERR, FR_OK, FATFS::free_clust, FS_FAT32, FATFS::fs_type, FATFS::fsi_flag, FSI_Free_Count, FSI_LeadSig, FSI_Nxt_Free, FATFS::fsi_sector, FSI_StrucSig, FATFS::last_clust, mem_set(), RES_OK, ST_DWORD, ST_WORD, sync_window(), FATFS::win, and FATFS::winsect.

Referenced by f_setlabel(), and f_sync().

4.11.2.33 static FRESULT sync_window (FATFS * *fs*) [static]

References disk_write(), FATFS::drv, FATFS::fatbase, FR_DISK_ERR, FR_OK, FATFS::fsize, FATFS::n_fats, RES_OK, FATFS::wflag, FATFS::win, and FATFS::winsect.

Referenced by dir_next(), f_write(), move_window(), and sync_fs().

4.11.2.34 static FRESULT validate (void * *obj*) [static]

References disk_status(), FATFS::drv, ENTER_FF, FR_INVALID_OBJECT, FR_NOT_READY, FR_OK, FIL::fs, FATFS::fs_type, FATFS::id, FIL::id, and STA_NOINIT.

Referenced by f_close(), f_lseek(), f_read(), f_sync(), and f_write().

4.11.3 Variable Documentation

4.11.3.1 const BYTE ExCvt[] = _EXCVT [static]

Referenced by create_name(), and f_setlabel().

4.11.3.2 FATFS* FatFs[_VOLUMES] [static]

4.11.3.3 WORD Fsid [static]

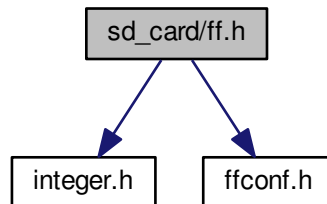
Referenced by chk_mounted().

4.12 sd_card/ff.h File Reference

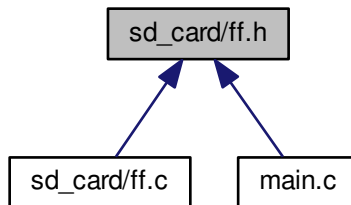
```
#include "integer.h"
```

```
#include "ffconf.h"
```

Include dependency graph for ff.h:



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



Data Structures

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

Macros

- `#define _FATFS 82786 /* Revision ID */`
- `#define LD2PD(vol) (BYTE)(vol) /* Each logical drive is bound to the same physical drive number */`
- `#define LD2PT(vol) 0 /* Always mounts the 1st partition or in SFD */`
- `#define _T(x) x`

- #define `_TEXT(x)` `x`
- #define `f_eof(fp)` `((fp)->fptr == (fp)->fsize) ? 1 : 0`
- #define `f_error(fp)` `((fp)->flag & FA__ERROR) ? 1 : 0`
- #define `f_tell(fp)` `((fp)->fptr)`
- #define `f_size(fp)` `((fp)->fsize)`
- #define `EOF` `(-1)`
- #define `FA_READ` `0x01`
- #define `FA_OPEN_EXISTING` `0x00`
- #define `FA__ERROR` `0x80`
- #define `FA_WRITE` `0x02`
- #define `FA_CREATE_NEW` `0x04`
- #define `FA_CREATE_ALWAYS` `0x08`
- #define `FA_OPEN_ALWAYS` `0x10`
- #define `FA__WRITTEN` `0x20`
- #define `FA__DIRTY` `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 `CREATE_LINKMAP` `0xFFFFFFFF`
- #define `LD_WORD(ptr)` `(WORD)*((WORD*)(BYTE*)(ptr))`
- #define `LD_DWORD(ptr)` `(DWORD)*((DWORD*)(BYTE*)(ptr))`
- #define `ST_WORD(ptr, val)` `*((WORD*)(BYTE*)(ptr))=(WORD)(val)`
- #define `ST_DWORD(ptr, val)` `*((DWORD*)(BYTE*)(ptr))=(DWORD)(val)`

Typedefs

- typedef char `TCHAR`

Enumerations

- enum `FRESULT` {
`FR_OK` = 0, `FR_DISK_ERR`, `FR_INT_ERR`, `FR_NOT_READY`,
`FR_NO_FILE`, `FR_NO_PATH`, `FR_INVALID_NAME`, `FR_DENIED`,
`FR_EXIST`, `FR_INVALID_OBJECT`, `FR_WRITE_PROTECTED`, `FR_INVALID_DRIVE`,
`FR_NOT_ENABLED`, `FR_NO_FILESYSTEM`, `FR_MKFS_ABORTED`, `FR_TIMEOUT`,
`FR_LOCKED`, `FR_NOT_ENOUGH_CORE`, `FR_TOO_MANY_OPEN_FILES`, `FR_INVALID_PARAMETER` }

Functions

- `FRESULT f_mount` (`BYTE` vol, `FATFS` *fs)
- `FRESULT f_open` (`FIL` *fp, const `TCHAR` *path, `BYTE` mode)
- `FRESULT f_read` (`FIL` *fp, void *buff, `UINT` btr, `UINT` *br)
- `FRESULT f_lseek` (`FIL` *fp, `DWORD` ofs)
- `FRESULT f_close` (`FIL` *fp)
- `FRESULT f_opendir` (`DIR` *dj, const `TCHAR` *path)
- `FRESULT f_readdir` (`DIR` *dj, `FILINFO` *fno)
- `FRESULT f_stat` (const `TCHAR` *path, `FILINFO` *fno)
- `FRESULT f_write` (`FIL` *fp, const void *buff, `UINT` btw, `UINT` *bw)
- `FRESULT f_getfree` (const `TCHAR` *path, `DWORD` *nclst, `FATFS` **fatfs)
- `FRESULT f_truncate` (`FIL` *fp)
- `FRESULT f_sync` (`FIL` *fp)
- `FRESULT f_unlink` (const `TCHAR` *path)
- `FRESULT f_mkdir` (const `TCHAR` *path)
- `FRESULT f_chmod` (const `TCHAR` *path, `BYTE` value, `BYTE` mask)
- `FRESULT f_utime` (const `TCHAR` *path, const `FILINFO` *fno)
- `FRESULT f_rename` (const `TCHAR` *path_old, const `TCHAR` *path_new)
- `FRESULT f_chdrive` (`BYTE` drv)
- `FRESULT f_chdir` (const `TCHAR` *path)
- `FRESULT f_getcwd` (`TCHAR` *buff, `UINT` len)
- `FRESULT f_getlabel` (const `TCHAR` *path, `TCHAR` *label, `DWORD` *sn)
- `FRESULT f_setlabel` (const `TCHAR` *label)
- `FRESULT f_forward` (`FIL` *fp, `UINT`(*func)(const `BYTE` *, `UINT`), `UINT` btf, `UINT` *bf)
- `FRESULT f_mkfs` (`BYTE` vol, `BYTE` sdf, `UINT` au)
- `FRESULT f_fdisk` (`BYTE` pdrv, const `DWORD` szt[], void *work)
- `int f_putc` (`TCHAR` c, `FIL` *fp)
- `int f_puts` (const `TCHAR` *str, `FIL` *cp)
- `int f_printf` (`FIL` *fp, const `TCHAR` *str,...)
- `TCHAR` * `f_gets` (`TCHAR` *buff, int len, `FIL` *fp)
- `DWORD get_fattime` (void)

4.12.1 Macro Definition Documentation

4.12.1.1 `#define _FATFS 82786 /* Revision ID */`

4.12.1.2 `#define _T(x) x`

4.12.1.3 `#define _TEXT(x) x`

4.12.1.4 `#define AM_ARC 0x20 /* Archive */`

Referenced by `f_sync()`.

4.12.1.5 `#define AM_DIR 0x10 /* Directory */`

Referenced by `f_open()`, and `follow_path()`.

4.12.1.6 `#define AM_HID 0x02 /* Hidden */`

4.12.1.7 `#define AM_LFN 0x0F /* LFN entry */`

Referenced by `dir_find()`, and `dir_read()`.

4.12.1.8 `#define AM_MASK 0x3F /* Mask of defined bits */`

Referenced by `dir_find()`, and `dir_read()`.

4.12.1.9 `#define AM_RDO 0x01 /* Read only */`

Referenced by `f_open()`.

4.12.1.10 `#define AM_SYS 0x04 /* System */`

4.12.1.11 `#define AM_VOL 0x08 /* Volume label */`

Referenced by `dir_find()`, `dir_read()`, and `f_setlabel()`.

4.12.1.12 `#define CREATE_LINKMAP 0xFFFFFFFF`

Referenced by `f_lseek()`.

4.12.1.13 `#define EOF (-1)`

4.12.1.14 `#define f_eof(fp) (((fp)->fptr == (fp)->fsize) ? 1 : 0)`

4.12.1.15 `#define f_error(fp) (((fp)->flag & FA__ERROR) ? 1 : 0)`

4.12.1.16 `#define f_size(fp) ((fp)->fsize)`

4.12.1.17 `#define f_tell(fp) ((fp)->fptr)`

4.12.1.18 `#define FA__DIRTY 0x40`

Referenced by `f_lseek()`, `f_read()`, `f_sync()`, and `f_write()`.

4.12.1.19 `#define FA__ERROR 0x80`

Referenced by `f_lseek()`, `f_read()`, and `f_write()`.

4.12.1.20 `#define FA__WRITTEN 0x20`

Referenced by `f_lseek()`, `f_open()`, `f_sync()`, and `f_write()`.

4.12.1.21 **#define FA_CREATE_ALWAYS 0x08**

Referenced by `f_open()`, and `main()`.

4.12.1.22 **#define FA_CREATE_NEW 0x04**

Referenced by `f_open()`.

4.12.1.23 **#define FA_OPEN_ALWAYS 0x10**

Referenced by `f_open()`.

4.12.1.24 **#define FA_OPEN_EXISTING 0x00**

4.12.1.25 **#define FA_READ 0x01**

Referenced by `f_open()`, `f_read()`, and `main()`.

4.12.1.26 **#define FA_WRITE 0x02**

Referenced by `f_lseek()`, `f_open()`, `f_write()`, and `main()`.

4.12.1.27 **#define FS_FAT12 1**

Referenced by `chk_mounted()`, `get_fat()`, and `put_fat()`.

4.12.1.28 **#define FS_FAT16 2**

Referenced by `chk_mounted()`, `get_fat()`, and `put_fat()`.

4.12.1.29 **#define FS_FAT32 3**

Referenced by `chk_mounted()`, `dir_sdi()`, `f_getlabel()`, `get_fat()`, `ld_clust()`, `put_fat()`, and `sync_fs()`.

4.12.1.30 **#define LD2PD(vol) (BYTE)(vol) /* Each logical drive is bound to the same physical drive number */**

Referenced by `chk_mounted()`.

4.12.1.31 **#define LD2PT(vol) 0 /* Always mounts the 1st partition or in SFD */**

Referenced by `chk_mounted()`.

4.12.1.32 **#define LD_DWORD(ptr) (DWORD)(*(DWORD*)(BYTE*)(ptr))**

Referenced by `check_fs()`, `chk_mounted()`, `f_getlabel()`, `f_open()`, `get_fat()`, and `put_fat()`.

4.12.1.33 `#define LD_WORD(ptr)(WORD)(*(WORD*)(BYTE*)(ptr))`

Referenced by `check_fs()`, `chk_mounted()`, `get_fat()`, and `ld_clust()`.

4.12.1.34 `#define ST_DWORD(ptr, val) *(DWORD*)(BYTE*)(ptr)=(DWORD)(val)`

Referenced by `f_open()`, `f_setlabel()`, `f_sync()`, `put_fat()`, and `sync_fs()`.

4.12.1.35 `#define ST_WORD(ptr, val) *(WORD*)(BYTE*)(ptr)=(WORD)(val)`

Referenced by `f_sync()`, `put_fat()`, `st_clust()`, and `sync_fs()`.

4.12.2 Typedef Documentation

4.12.2.1 `typedef char TCHAR`

4.12.3 Enumeration Type Documentation

4.12.3.1 `enum FRESULT`

Enumerator

FR_OK
FR_DISK_ERR
FR_INT_ERR
FR_NOT_READY
FR_NO_FILE
FR_NO_PATH
FR_INVALID_NAME
FR_DENIED
FR_EXIST
FR_INVALID_OBJECT
FR_WRITE_PROTECTED
FR_INVALID_DRIVE
FR_NOT_ENABLED
FR_NO_FILESYSTEM
FR_MKFS_ABORTED
FR_TIMEOUT
FR_LOCKED
FR_NOT_ENOUGH_CORE
FR_TOO_MANY_OPEN_FILES
FR_INVALID_PARAMETER

4.12.4 Function Documentation

4.12.4.1 **FRESULT** `f_chdir (const TCHAR * path)`

4.12.4.2 **FRESULT** `f_chdrive (BYTE drv)`

4.12.4.3 **FRESULT** `f_chmod (const TCHAR * path, BYTE value, BYTE mask)`

4.12.4.4 **FRESULT** `f_close (FIL * fp)`

References `f_sync()`, `FR_OK`, `FIL::fs`, `LEAVE_FF`, and `validate()`.

Referenced by `main()`.

4.12.4.5 **FRESULT** `f_fdisk (BYTE pdrv, const DWORD szf[], void * work)`

4.12.4.6 **FRESULT** `f_forward (FIL * fp, UINT(*) (const BYTE *, UINT) func, UINT btf, UINT * bf)`

4.12.4.7 **FRESULT** `f_getcwd (TCHAR * buff, UINT len)`

4.12.4.8 **FRESULT** `f_getfree (const TCHAR * path, DWORD * nclst, FATFS * * fatfs)`

4.12.4.9 **FRESULT** `f_getlabel (const TCHAR * path, TCHAR * label, DWORD * sn)`

References `BS_VolID`, `BS_VolID32`, `chk_mounted()`, `DIR::dir`, `dir_read()`, `dir_sdi()`, `FR_NO_FILE`, `FR_OK`, `DIR::fs`, `F-S_FAT32`, `FATFS::fs_type`, `IsDBCS1`, `IsDBCS2`, `LD_DWORD`, `LEAVE_FF`, `mem_cpy()`, `move_window()`, `DIR::sclust`, `FATFS::volbase`, and `FATFS::win`.

Referenced by `main()`.

4.12.4.10 **TCHAR*** `f_gets (TCHAR * buff, int len, FIL * fp)`

4.12.4.11 **FRESULT** `f_lseek (FIL * fp, DWORD ofs)`

References `_FS_READONLY`, `ABORT`, `FIL::clust`, `clust2sect()`, `create_chain()`, `CREATE_LINKMAP`, `FATFS::csize`, `disk_read()`, `disk_write()`, `FATFS::drv`, `FIL::dsect`, `FA_DIRTY`, `FA_ERROR`, `FA_WRITTEN`, `FA_WRITE`, `FIL::flag`, `FIL::fptr`, `FR_DISK_ERR`, `FR_INT_ERR`, `FR_NOT_ENOUGH_CORE`, `FR_OK`, `FIL::fs`, `FIL::fsize`, `get_fat()`, `LEAVE_FF`, `FATFS::n_fatent`, `RES_OK`, `FIL::sclust`, `SS`, and `validate()`.

4.12.4.12 **FRESULT** `f_mkdir (const TCHAR * path)`

4.12.4.13 **FRESULT** `f_mkfs (BYTE vol, BYTE sfd, UINT au)`

4.12.4.14 **FRESULT** `f_mount (BYTE vol, FATFS * fs)`

References `_VOLUMES`, `FR_INT_ERR`, `FR_INVALID_DRIVE`, `FR_OK`, and `FATFS::fs_type`.

Referenced by `main()`.

4.12.4.15 FRESULT f_open (FIL * *fp*, const TCHAR * *path*, BYTE *mode*)

References AM_DIR, AM_RDO, chk_mounted(), DEF_NAMEBUF, DIR::dir, DIR_Attr, DIR_CrtTime, DIR_FileSize, FIL::dir_ptr, dir_register(), FIL::dir_sect, FIL::dsect, FA__WRITTEN, FA_CREATE_ALWAYS, FA_CREATE_NEW, FA_OPEN_ALWAYS, FA_READ, FA_WRITE, FIL::flag, follow_path(), FIL::fptr, FR_DENIED, FR_EXIST, FR_INT_ERR, FR_INVALID_NAME, FR_INVALID_OBJECT, FR_NO_FILE, FR_OK, FR_TOO_MANY_OPEN_FILES, FREE_BUF, FIL::fs, DIR::fs, FIL::fsize, get_fattime(), FATFS::id, FIL::id, INIT_BUF, FATFS::last_clust, Id_clust(), LD_DWORD, LEAVE_FF, move_window(), remove_chain(), FIL::sclust, st_clust(), ST_DWORD, FATFS::wflag, and FATFS::winsect.

Referenced by main().

4.12.4.16 FRESULT f_opendir (DIR * *dj*, const TCHAR * *path*)**4.12.4.17 int f_printf (FIL * *fp*, const TCHAR * *str*, ...)****4.12.4.18 int f_putc (TCHAR *c*, FIL * *fp*)****4.12.4.19 int f_puts (const TCHAR * *str*, FIL * *cp*)****4.12.4.20 FRESULT f_read (FIL * *fp*, void * *buff*, UINT *btr*, UINT * *br*)**

References ABORT, FIL::clust, clust2sect(), FATFS::csize, disk_read(), disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__ERROR, FA_READ, FIL::flag, FIL::fptr, FR_DENIED, FR_DISK_ERR, FR_INT_ERR, FR_OK, FIL::fs, FIL::fsize, get_fat(), LEAVE_FF, mem_cpy(), move_window(), RES_OK, FIL::sclust, SS, validate(), FATFS::wflag, FATFS::win, and FATFS::winsect.

Referenced by main().

4.12.4.21 FRESULT f_readdir (DIR * *dj*, FILINFO * *fno*)**4.12.4.22 FRESULT f_rename (const TCHAR * *path_old*, const TCHAR * *path_new*)****4.12.4.23 FRESULT f_setlabel (const TCHAR * *label*)**

References _DF1S, AM_VOL, chk_chr(), chk_mounted(), DDE, DIR::dir, dir_alloc(), DIR_Attr, dir_read(), dir_sdi(), DIR_WrtTime, ExCvt, FR_INVALID_NAME, FR_NO_FILE, FR_OK, DIR::fs, get_fattime(), IsDBCS1, IsDBCS2, IsLower, LEAVE_FF, mem_cpy(), mem_set(), DIR::sclust, ST_DWORD, sync_fs(), SZ_DIR, and FATFS::wflag.

4.12.4.24 FRESULT f_stat (const TCHAR * *path*, FILINFO * *fno*)**4.12.4.25 FRESULT f_sync (FIL * *fp*)**

References AM_ARC, DIR_Attr, DIR_FileSize, DIR_LstAccDate, FIL::dir_ptr, FIL::dir_sect, DIR_WrtTime, disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__WRITTEN, FIL::flag, FR_DISK_ERR, FR_OK, FIL::fs, FIL::fsize, get_fattime(), LEAVE_FF, move_window(), RES_OK, FIL::sclust, st_clust(), ST_DWORD, ST_WORD, sync_fs(), validate(), and FATFS::wflag.

Referenced by f_close().

4.12.4.26 FRESULT f_truncate (FIL * *fp*)

4.12.4.27 **FRESULT** `f_unlink (const TCHAR * path)`

4.12.4.28 **FRESULT** `f_ftime (const TCHAR * path, const FILINFO * fno)`

4.12.4.29 **FRESULT** `f_write (FIL * fp, const void * buff, UINT btw, UINT * bw)`

References ABORT, FIL::clust, clust2sect(), create_chain(), FATFS::csize, disk_read(), disk_write(), FATFS::drv, FIL::dsect, FA__DIRTY, FA__ERROR, FA__WRITTEN, FA_WRITE, FIL::flag, FIL::fptr, FR_DENIED, FR_DISK_ERR, FR_INT_ERR, FR_OK, FIL::fs, FIL::fsize, LEAVE_FF, mem_cpy(), move_window(), RES_OK, FIL::sclust, SS, sync_window(), validate(), FATFS::wflag, FATFS::win, and FATFS::winsect.

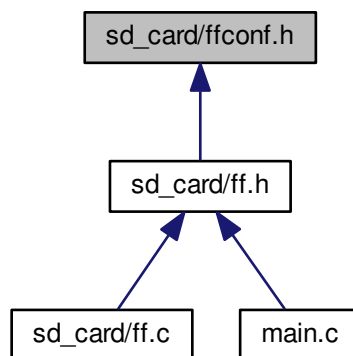
Referenced by `main()`.

4.12.4.30 **DWORD** `get_fattime (void)`

Referenced by `f_open()`, `f_setlabel()`, and `f_sync()`.

4.13 sd_card/ffconf.h File Reference

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



Macros

- `#define _FFCONF 82786 /* Revision ID */`
- `#define _FS_TINY 1 /* 0:Normal or 1:Tiny */`
- `#define _FS_READONLY 0 /* 0:Read/Write or 1:Read only */`
- `#define _FS_MINIMIZE 2 /* 0 to 3 */`
- `#define _USE_STRFUNC 0 /* 0:Disable or 1-2:Enable */`
- `#define _USE_MKFS 0 /* 0:Disable or 1:Enable */`
- `#define _USE_FASTSEEK 0 /* 0:Disable or 1:Enable */`
- `#define _USE_LABEL 1 /* 0:Disable or 1:Enable */`

- `#define _USE_FORWARD 0 /* 0:Disable or 1:Enable */`
- `#define _CODE_PAGE 437`
- `#define _USE_LFN 0 /* 0 to 3 */`
- `#define _MAX_LFN 255 /* Maximum LFN length to handle (12 to 255) */`
- `#define _LFN_UNICODE 0 /* 0:ANSI/OEM or 1:Unicode */`
- `#define _FS_RPATH 0 /* 0 to 2 */`
- `#define _VOLUMES 1`
- `#define _MAX_SS 512 /* 512, 1024, 2048 or 4096 */`
- `#define _MULTI_PARTITION 0 /* 0:Single partition, 1:Enable multiple partition */`
- `#define _USE_ERASE 0 /* 0:Disable or 1:Enable */`
- `#define _WORD_ACCESS 1 /* 0 or 1 */`
- `#define _FS_REENTRANT 0 /* 0:Disable or 1:Enable */`
- `#define _FS_TIMEOUT 1000 /* Timeout period in unit of time ticks */`
- `#define _SYNC_t HANDLE /* O/S dependent type of sync object. e.g. HANDLE, OS_EVENT*, ID and etc.. */`
- `#define _FS_LOCK 0 /* 0:Disable or >=1:Enable */`

4.13.1 Macro Definition Documentation

4.13.1.1 `#define _CODE_PAGE 437`

4.13.1.2 `#define _FFCONF 82786 /* Revision ID */`

4.13.1.3 `#define _FS_LOCK 0 /* 0:Disable or >=1:Enable */`

4.13.1.4 `#define _FS_MINIMIZE 2 /* 0 to 3 */`

4.13.1.5 `#define _FS_READONLY 0 /* 0:Read/Write or 1:Read only */`

Referenced by `chk_mounted()`, and `f_lseek()`.

4.13.1.6 `#define _FS_REENTRANT 0 /* 0:Disable or 1:Enable */`

4.13.1.7 `#define _FS_RPATH 0 /* 0 to 2 */`

Referenced by `dir_read()`, `dir_register()`, and `follow_path()`.

4.13.1.8 `#define _FS_TIMEOUT 1000 /* Timeout period in unit of time ticks */`

4.13.1.9 `#define _FS_TINY 1 /* 0:Normal or 1:Tiny */`

4.13.1.10 `#define _LFN_UNICODE 0 /* 0:ANSI/OEM or 1:Unicode */`

4.13.1.11 `#define _MAX_LFN 255 /* Maximum LFN length to handle (12 to 255) */`

Referenced by `create_name()`.

4.13.1.12 `#define _MAX_SS 512 /* 512, 1024, 2048 or 4096 */`

4.13.1.13 `#define _MULTI_PARTITION 0 /* 0:Single partition, 1:Enable multiple partition */`

4.13.1.14 `#define _SYNC_t HANDLE /* O/S dependent type of sync object. e.g. HANDLE, OS_EVENT*, ID and etc.. */`

4.13.1.15 `#define _USE_ERASE 0 /* 0:Disable or 1:Enable */`

4.13.1.16 `#define _USE_FASTSEEK 0 /* 0:Disable or 1:Enable */`

4.13.1.17 `#define _USE_FORWARD 0 /* 0:Disable or 1:Enable */`

4.13.1.18 `#define _USE_LABEL 1 /* 0:Disable or 1:Enable */`

4.13.1.19 `#define _USE_LFN 0 /* 0 to 3 */`

4.13.1.20 `#define _USE_MKFS 0 /* 0:Disable or 1:Enable */`

4.13.1.21 `#define _USE_STRFUNC 0 /* 0:Disable or 1-2:Enable */`

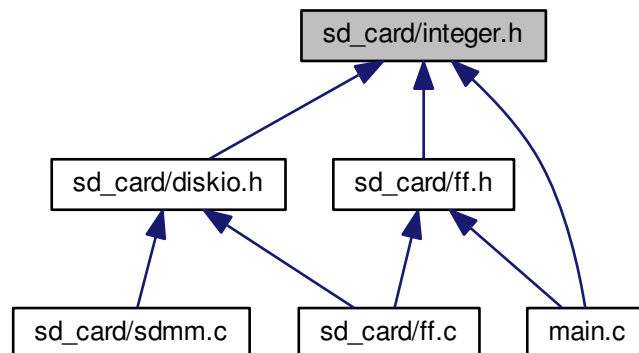
4.13.1.22 `#define _VOLUMES 1`

Referenced by `chk_mounted()`, and `f_mount()`.

4.13.1.23 `#define _WORD_ACCESS 1 /* 0 or 1 */`

4.14 sd_card/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](#)

4.14.1 Typedef Documentation

4.14.1.1 typedef unsigned char [BYTE](#)

4.14.1.2 typedef char [CHAR](#)

4.14.1.3 typedef unsigned long [DWORD](#)

4.14.1.4 typedef int [INT](#)

4.14.1.5 typedef long [LONG](#)

4.14.1.6 typedef short [SHORT](#)

4.14.1.7 typedef unsigned char [UCHAR](#)

4.14.1.8 typedef unsigned int [UINT](#)

4.14.1.9 typedef unsigned long [ULONG](#)

4.14.1.10 typedef unsigned short [USHORT](#)

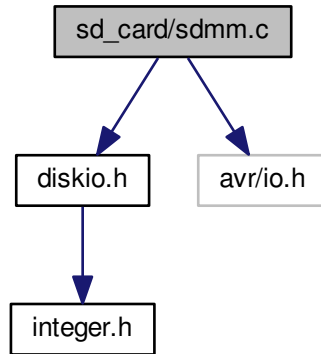
4.14.1.11 typedef unsigned short [WCHAR](#)

4.14.1.12 typedef unsigned short [WORD](#)

4.15 sd_card/sdmm.c File Reference

```
#include "diskio.h"  
#include <avr/io.h>
```

Include dependency graph for sdmm.c:



Macros

- `#define SD_COM_PORT PORTA`
- `#define DO_INIT() /* Initialize port MMC DO as input */`
- `#define DO_DQ 3`
- `#define DO (SD_COM_PORT.IN & (1<<DO_DQ)) /* Test for MMC DO ('H':true, 'L':false) */`
- `#define DI_DQ 6`
- `#define DI_INIT() SD_COM_PORT.DIRSET |= (1<<DI_DQ) /* Initialize port MMC DI as output */`
- `#define DI_H() SD_COM_PORT.OUT |= (1<<DI_DQ) /* Set MMC DI "high" */`
- `#define DI_L() SD_COM_PORT.OUT &= ~(1<<DI_DQ) /* Set MMC DI "low" */`
- `#define CK_DQ 4`
- `#define CK_INIT() SD_COM_PORT.DIRSET |= (1<<CK_DQ) /* Initialize port MMC SCLK as output */`
- `#define CK_H() SD_COM_PORT.OUT |= (1<<CK_DQ) /* Set MMC SCLK "high" */`
- `#define CK_L() SD_COM_PORT.OUT &= ~(1<<CK_DQ) /* Set MMC SCLK "low" */`
- `#define CS_DQ 5`
- `#define CS_INIT() SD_COM_PORT.DIRSET |= (1<<CS_DQ) /* Initialize port MMC CS as output */`
- `#define CS_H() SD_COM_PORT.OUT |= (1<<CS_DQ) /* Set MMC CS "high" */`
- `#define CS_L() SD_COM_PORT.OUT &= ~(1<<CS_DQ) /* Set MMC CS "low" */`
- `#define CMD0 (0) /* GO_IDLE_STATE */`
- `#define CMD1 (1) /* SEND_OP_COND */`
- `#define ACMD41 (0x80+41) /* SEND_OP_COND (SDC) */`
- `#define CMD8 (8) /* SEND_IF_COND */`
- `#define CMD9 (9) /* SEND_CSD */`
- `#define CMD10 (10) /* SEND_CID */`
- `#define CMD12 (12) /* STOP_TRANSMISSION */`
- `#define CMD13 (13) /* SEND_STATUS */`
- `#define ACMD13 (0x80+13) /* SD_STATUS (SDC) */`
- `#define CMD16 (16) /* SET_BLOCKLEN */`
- `#define CMD17 (17) /* READ_SINGLE_BLOCK */`
- `#define CMD18 (18) /* READ_MULTIPLE_BLOCK */`

- `#define CMD23 (23) /* SET_BLOCK_COUNT */`
- `#define ACMD23 (0x80+23) /* SET_WR_BLK_ERASE_COUNT (SDC) */`
- `#define CMD24 (24) /* WRITE_BLOCK */`
- `#define CMD25 (25) /* WRITE_MULTIPLE_BLOCK */`
- `#define CMD32 (32) /* ERASE_ER_BLK_START */`
- `#define CMD33 (33) /* ERASE_ER_BLK_END */`
- `#define CMD38 (38) /* ERASE */`
- `#define CMD55 (55) /* APP_CMD */`
- `#define CMD58 (58) /* READ_OCR */`

Functions

- static void `dly_us` (UINT n)
- static void `xmit_mmc` (const BYTE *buff, UINT bc)
- static void `rcvr_mmc` (BYTE *buff, UINT bc)
- static int `wait_ready` (void)
- static void `deselect` (void)
- static int `select` (void)
- static int `rcvr_datablock` (BYTE *buff, UINT btr)
- static int `xmit_datablock` (const BYTE *buff, BYTE token)
- static BYTE `send_cmd` (BYTE cmd, DWORD arg)
- `DSTATUS` `disk_status` (BYTE drv)
- `DSTATUS` `disk_initialize` (BYTE drv)
- `DRESULT` `disk_read` (BYTE drv, BYTE *buff, DWORD sector, BYTE count)
- `DRESULT` `disk_write` (BYTE drv, const BYTE *buff, DWORD sector, BYTE count)
- `DRESULT` `disk_ioctl` (BYTE drv, BYTE ctrl, void *buff)

Variables

- static `DSTATUS` `Stat` = `STA_NOINIT`
- static BYTE `CardType`

4.15.1 Macro Definition Documentation

4.15.1.1 `#define ACMD13 (0x80+13) /* SD_STATUS (SDC) */`

4.15.1.2 `#define ACMD23 (0x80+23) /* SET_WR_BLK_ERASE_COUNT (SDC) */`

Referenced by `disk_write()`.

4.15.1.3 `#define ACMD41 (0x80+41) /* SEND_OP_COND (SDC) */`

Referenced by `disk_initialize()`.

4.15.1.4 `#define CK_DQ 4`

4.15.1.5 `#define CK_H() SD_COM_PORT.OUT |= (1<<CK_DQ) /* Set MMC SCLK "high" */`

Referenced by `rcvr_mmc()`, and `xmit_mmc()`.

4.15.1.6 `#define CK_INIT() SD_COM_PORT.DIRSET |= (1<<CK_DQ) /* Initialize port MMC SCLK as output */`

Referenced by `disk_initialize()`.

4.15.1.7 `#define CK_L() SD_COM_PORT.OUT &= ~(1<<CK_DQ) /* Set MMC SCLK "low" */`

Referenced by `disk_initialize()`, `rcvr_mmc()`, and `xmit_mmc()`.

4.15.1.8 `#define CMD0 (0) /* GO_IDLE_STATE */`

Referenced by `disk_initialize()`, and `send_cmd()`.

4.15.1.9 `#define CMD1 (1) /* SEND_OP_COND */`

Referenced by `disk_initialize()`.

4.15.1.10 `#define CMD10 (10) /* SEND_CID */`

4.15.1.11 `#define CMD12 (12) /* STOP_TRANSMISSION */`

Referenced by `disk_read()`, and `send_cmd()`.

4.15.1.12 `#define CMD13 (13) /* SEND_STATUS */`

4.15.1.13 `#define CMD16 (16) /* SET_BLOCKLEN */`

Referenced by `disk_initialize()`.

4.15.1.14 `#define CMD17 (17) /* READ_SINGLE_BLOCK */`

Referenced by `disk_read()`.

4.15.1.15 `#define CMD18 (18) /* READ_MULTIPLE_BLOCK */`

Referenced by `disk_read()`.

4.15.1.16 `#define CMD23 (23) /* SET_BLOCK_COUNT */`

4.15.1.17 `#define CMD24 (24) /* WRITE_BLOCK */`

Referenced by `disk_write()`.

4.15.1.18 `#define CMD25 (25) /* WRITE_MULTIPLE_BLOCK */`

Referenced by `disk_write()`.

4.15.1.19 `#define CMD32 (32) /* ERASE_ER_BLK_START */`

4.15.1.20 `#define CMD33 (33) /* ERASE_ER_BLK_END */`

4.15.1.21 `#define CMD38 (38) /* ERASE */`

4.15.1.22 `#define CMD55 (55) /* APP_CMD */`

Referenced by `send_cmd()`.

4.15.1.23 `#define CMD58 (58) /* READ_OCR */`

Referenced by `disk_initialize()`.

4.15.1.24 `#define CMD8 (8) /* SEND_IF_COND */`

Referenced by `disk_initialize()`, and `send_cmd()`.

4.15.1.25 `#define CMD9 (9) /* SEND_CSD */`

Referenced by `disk_ioctl()`.

4.15.1.26 `#define CS_DQ 5`

4.15.1.27 `#define CS_H() SD_COM_PORT.OUT |= (1<<CS_DQ) /* Set MMC CS "high" */`

Referenced by `deselect()`, and `disk_initialize()`.

4.15.1.28 `#define CS_INIT() SD_COM_PORT.DIRSET |= (1<<CS_DQ) /* Initialize port MMC CS as output */`

Referenced by `disk_initialize()`.

4.15.1.29 `#define CS_L() SD_COM_PORT.OUT &= ~(1<<CS_DQ) /* Set MMC CS "low" */`

Referenced by `select()`.

4.15.1.30 `#define DI_DQ 6`

4.15.1.31 `#define DI_H() SD_COM_PORT.OUT |= (1<<DI_DQ) /* Set MMC DI "high" */`

Referenced by `rcvr_mmc()`, and `xmit_mmc()`.

4.15.1.32 `#define DI_INIT() SD_COM_PORT.DIRSET |= (1<<DI_DQ) /* Initialize port MMC DI as output */`

Referenced by `disk_initialize()`.

4.15.1.33 **#define** DI_L() SD_COM_PORT.OUT &= ~(1<<DI_DQ) /* Set MMC DI "low" */

Referenced by xmit_mmc().

4.15.1.34 **#define** DO (SD_COM_PORT.IN & (1<<DO_DQ)) /* Test for MMC DO ('H':true, 'L':false) */

Referenced by rcvr_mmc().

4.15.1.35 **#define** DO_DQ 3

4.15.1.36 **#define** DO_INIT() /* Initialize port MMC DO as input */

Referenced by disk_initialize().

4.15.1.37 **#define** SD_COM_PORT PORTA

Referenced by dly_us().

4.15.2 Function Documentation

4.15.2.1 **static void** deselect (void) [static]

References CS_H, and rcvr_mmc().

Referenced by disk_initialize(), disk_ioctl(), disk_read(), disk_write(), select(), and send_cmd().

4.15.2.2 **DSTATUS** disk_initialize (BYTE drv)

References ACMD41, CardType, CK_INIT, CK_L, CMD0, CMD1, CMD16, CMD58, CMD8, CS_H, CS_INIT, CT_BLOCK, CT_MMC, CT_SD1, CT_SD2, deselect(), DI_INIT, dly_us(), DO_INIT, rcvr_mmc(), RES_NOTRDY, send_cmd(), STA_NOINIT, and Stat.

Referenced by chk_mounted().

4.15.2.3 **DRESULT** disk_ioctl (BYTE drv, BYTE ctrl, void * buff)

References CMD9, CTRL_SYNC, deselect(), disk_status(), GET_BLOCK_SIZE, GET_SECTOR_COUNT, rcvr_datablock(), RES_ERROR, RES_NOTRDY, RES_OK, RES_PARERR, select(), send_cmd(), and STA_NOINIT.

Referenced by chk_mounted(), remove_chain(), and sync_fs().

4.15.2.4 **DRESULT** disk_read (BYTE drv, BYTE * buff, DWORD sector, BYTE count)

References CardType, CMD12, CMD17, CMD18, CT_BLOCK, deselect(), disk_status(), rcvr_datablock(), RES_ERROR, RES_NOTRDY, RES_OK, send_cmd(), and STA_NOINIT.

Referenced by check_fs(), chk_mounted(), f_lseek(), f_read(), f_write(), and move_window().

4.15.2.5 **DSTATUS** disk_status (*BYTE drv*)

References STA_NOINIT, and Stat.

Referenced by chk_mounted(), disk_ioctl(), disk_read(), disk_write(), and validate().

4.15.2.6 **DRESULT** disk_write (*BYTE drv*, *const BYTE * buff*, *DWORD sector*, *BYTE count*)

References ACMD23, CardType, CMD24, CMD25, CT_BLOCK, CT_SDC, deselect(), disk_status(), RES_ERROR, RES_NOTRDY, RES_OK, send_cmd(), STA_NOINIT, and xmit_datablock().

Referenced by f_lseek(), f_read(), f_sync(), f_write(), sync_fs(), and sync_window().

4.15.2.7 **static void** dly_us (*UINT n*) [static]

References SD_COM_PORT.

Referenced by disk_initialize(), rcvr_datablock(), and wait_ready().

4.15.2.8 **static int** rcvr_datablock (*BYTE * buff*, *UINT btr*) [static]

References dly_us(), and rcvr_mmc().

Referenced by disk_ioctl(), and disk_read().

4.15.2.9 **static void** rcvr_mmc (*BYTE * buff*, *UINT bc*) [static]

References CK_H, CK_L, DI_H, and DO.

Referenced by deselect(), disk_initialize(), rcvr_datablock(), select(), send_cmd(), wait_ready(), and xmit_datablock().

4.15.2.10 **static int** select (*void*) [static]

References CS_L, deselect(), rcvr_mmc(), and wait_ready().

Referenced by disk_ioctl(), and send_cmd().

4.15.2.11 **static BYTE** send_cmd (*BYTE cmd*, *DWORD arg*) [static]

References CMD0, CMD12, CMD55, CMD8, deselect(), rcvr_mmc(), select(), and xmit_mmc().

Referenced by disk_initialize(), disk_ioctl(), disk_read(), and disk_write().

4.15.2.12 **static int** wait_ready (*void*) [static]

References dly_us(), and rcvr_mmc().

Referenced by select(), and xmit_datablock().

4.15.2.13 **static int** xmit_datablock (*const BYTE * buff*, *BYTE token*) [static]

References rcvr_mmc(), wait_ready(), and xmit_mmc().

Referenced by `disk_write()`.

4.15.2.14 `static void xmit_mmc (const BYTE * buff, UINT bc) [static]`

References `CK_H`, `CK_L`, `DI_H`, and `DI_L`.

Referenced by `send_cmd()`, and `xmit_datablock()`.

4.15.3 Variable Documentation

4.15.3.1 `BYTE CardType [static]`

Referenced by `disk_initialize()`, `disk_read()`, and `disk_write()`.

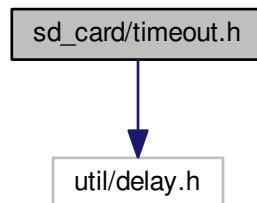
4.15.3.2 `DSTATUS Stat = STA_NOINIT [static]`

Referenced by `disk_initialize()`, and `disk_status()`.

4.16 sd_card/timeout.h File Reference

```
#include <util/delay.h>
```

Include dependency graph for `timeout.h`:



Macros

- `#define F_CPU 16000000UL`

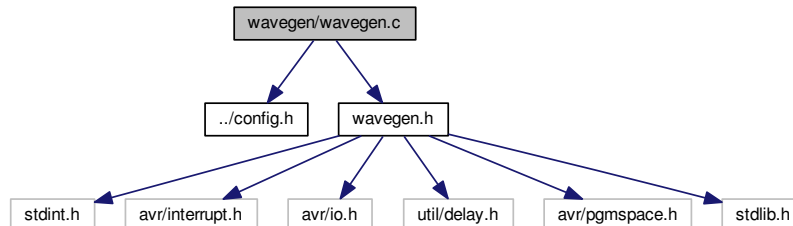
4.16.1 Macro Definition Documentation

4.16.1.1 `#define F_CPU 16000000UL`

Referenced by `wavegen_setFrequency()`, and `wavegen_setFrequency2()`.

4.17 wavegen/wavegen.c File Reference

```
#include "../config.h"
#include "wavegen.h"
Include dependency graph for wavegen.c:
```



Functions

- [ISR](#) (TCC0_CCA_vect)
- [ISR](#) (TCC0_CCB_vect)
- void [wavegen_setSound](#) (char startStop, char bankNum)
- void [wavegen_disableSound](#) (char bankNum)
- void [wavegen_sineWave](#) (char wavetableNum)
- void [wavegen_noiseWave](#) (char wavetableNum)
- void [wavegen_pwmInit](#) ()
- void [wavegen_pwmSet](#) (char channel, int value)
- void [wavegen_setFrequency](#) (unsigned int freq)
- void [wavegen_setFrequency2](#) (unsigned int freq)
- void [wavegen_clock_init](#) (void)

Variables

- unsigned char [wavetable](#) [256]
- unsigned char [wavetable2](#) [256]
- unsigned char [wavetable3](#) [256]
- const unsigned char [sinetable](#)[256] [PROGMEM](#)
- unsigned int [frequencyCoef](#) =100
- unsigned int [frequencyCoef2](#) =100
- char [sound1Enabled](#) =0
- char [sound2Enabled](#) =0
- char [sound3Enabled](#) =0
- char [soundPWM](#) =1
- int [currentVoice](#) =0

4.17.1 Function Documentation

4.17.1.1 `ISR (TCC0_CCA_vect)`

References `frequencyCoef`, `sound1Enabled`, `soundPWM`, `wavegen_pwmSet()`, and `wavetable`.

4.17.1.2 `ISR (TCC0_CCB_vect)`

References `frequencyCoef2`, `sound2Enabled`, `soundPWM`, `wavegen_pwmSet()`, and `wavetable2`.

4.17.1.3 `void wavegen_clock_init (void)`

Referenced by `main()`.

4.17.1.4 `void wavegen_disableSound (char bankNum)`

References `sound1Enabled`, `sound2Enabled`, and `sound3Enabled`.

4.17.1.5 `void wavegen_noiseWave (char wavetableNum)`

References `wavetable`, `wavetable2`, and `wavetable3`.

Referenced by `main()`.

4.17.1.6 `void wavegen_pwmInit ()`

Referenced by `main()`.

4.17.1.7 `void wavegen_pwmSet (char channel, int value)`

Referenced by `ISR()`.

4.17.1.8 `void wavegen_setFrequency (unsigned int freq)`

References `F_CPU`, `frequencyCoef`, `FS`, and `soundPWM`.

Referenced by `main()`.

4.17.1.9 `void wavegen_setFrequency2 (unsigned int freq)`

References `F_CPU`, `frequencyCoef2`, `FS`, and `soundPWM`.

Referenced by `main()`.

4.17.1.10 `void wavegen_setSound (char startStop, char bankNum)`

References `sound1Enabled`, `sound2Enabled`, and `sound3Enabled`.

Referenced by `main()`.

4.17.1.11 void wavegen_sineWave (char wavetableNum)

References wavetable, wavetable2, and wavetable3.

Referenced by main().

4.17.2 Variable Documentation

4.17.2.1 int currentVoice =0

4.17.2.2 unsigned int frequencyCoef =100

Referenced by ISR(), and wavegen_setFrequency().

4.17.2.3 unsigned int frequencyCoef2 =100

Referenced by ISR(), and wavegen_setFrequency2().

4.17.2.4 const unsigned char sinetable [256] PROGMEM

Initial value:

```
= {
128,131,134,137,140,143,146,149,152,156,159,162,165,168,171,174,
176,179,182,185,188,191,193,196,199,201,204,206,209,211,213,216,
218,220,222,224,226,228,230,232,234,236,237,239,240,242,243,245,
246,247,248,249,250,251,252,252,253,254,254,255,255,255,255,
255,255,255,255,255,255,254,254,253,252,252,251,250,249,248,247,
246,245,243,242,240,239,237,236,234,232,230,228,226,224,222,220,
218,216,213,211,209,206,204,201,199,196,193,191,188,185,182,179,
176,174,171,168,165,162,159,156,152,149,146,143,140,137,134,131,
128,124,121,118,115,112,109,106,103,99, 96, 93, 90, 87, 84, 81,
79, 76, 73, 70, 67, 64, 62, 59, 56, 54, 51, 49, 46, 44, 42, 39,
37, 35, 33, 31, 29, 27, 25, 23, 21, 19, 18, 16, 15, 13, 12, 10,
9, 8, 7, 6, 5, 4, 3, 3, 2, 1, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 2, 3, 3, 4, 5, 6, 7, 8,
9, 10, 12, 13, 15, 16, 18, 19, 21, 23, 25, 27, 29, 31, 33, 35,
37, 39, 42, 44, 46, 49, 51, 54, 56, 59, 62, 64, 67, 70, 73, 76,
79, 81, 84, 87, 90, 93, 96, 99, 103,106,109,112,115,118,121,124
}
```

4.17.2.5 char sound1Enabled =0

Referenced by ISR(), wavegen_disableSound(), and wavegen_setSound().

4.17.2.6 char sound2Enabled =0

Referenced by ISR(), wavegen_disableSound(), and wavegen_setSound().

4.17.2.7 char sound3Enabled =0

Referenced by wavegen_disableSound(), and wavegen_setSound().

4.17.2.8 char soundPWM =1

Referenced by ISR(), wavegen_setFrequency(), and wavegen_setFrequency2().

4.17.2.9 unsigned char wavetable[256]

Referenced by ISR(), wavegen_noiseWave(), and wavegen_sineWave().

4.17.2.10 unsigned char wavetable2[256]

Referenced by ISR(), wavegen_noiseWave(), and wavegen_sineWave().

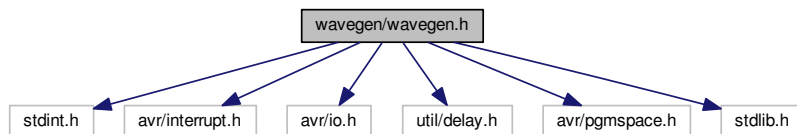
4.17.2.11 unsigned char wavetable3[256]

Referenced by wavegen_noiseWave(), and wavegen_sineWave().

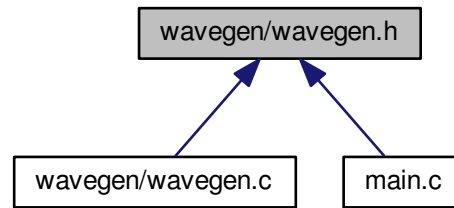
4.18 wavegen/wavegen.h File Reference

```
#include <stdint.h>
#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include <stdlib.h>
```

Include dependency graph for wavegen.h:



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



Macros

- `#define INTERRUPT_PERIOD 512`
- `#define FINT (F_CPU / INTERRUPT_PERIOD)`
- `#define FS (FINT)`

Functions

- void `wavegen_pwmInit ()`
- void `wavegen_clock_init (void)`
- void `wavegen_pwmSet (char channel, int value)`
- void `wavegen_sineWave (char wavetableNum)`
- void `wavegen_noiseWave (char wavetableNum)`
- void `wavegen_setFrequency2 ()`
- void `wavegen_setFrequency ()`
- void `wavegen_setSound (char startStop, char bankNum)`
- void `wavegen_disableSound (char bankNum)`

4.18.1 Macro Definition Documentation

4.18.1.1 `#define FINT (F_CPU / INTERRUPT_PERIOD)`

4.18.1.2 `#define FS (FINT)`

Referenced by `wavegen_setFrequency()`, and `wavegen_setFrequency2()`.

4.18.1.3 `#define INTERRUPT_PERIOD 512`

4.18.2 Function Documentation

4.18.2.1 void `wavegen_clock_init (void)`

Referenced by `main()`.

4.18.2.2 void wavegen_disableSound (char *bankNum*)

References sound1Enabled, sound2Enabled, and sound3Enabled.

4.18.2.3 void wavegen_noiseWave (char *wavetableNum*)

References wavetable, wavetable2, and wavetable3.

Referenced by main().

4.18.2.4 void wavegen_pwmInit ()

Referenced by main().

4.18.2.5 void wavegen_pwmSet (char *channel*, int *value*)

Referenced by ISR().

4.18.2.6 void wavegen_setFrequency ()**4.18.2.7 void wavegen_setFrequency2 ()****4.18.2.8 void wavegen_setSound (char *startStop*, char *bankNum*)**

References sound1Enabled, sound2Enabled, and sound3Enabled.

Referenced by main().

4.18.2.9 void wavegen_sineWave (char *wavetableNum*)

References wavetable, wavetable2, and wavetable3.

Referenced by main().

Index

`_CODE_PAGE`
 ffconf.h, 63

`_DF1S`
 ff.c, 42

`_EXCVT`
 ff.c, 42

`_FATFS`
 ff.h, 56

`_FFCONF`
 ffconf.h, 63

`_FS_LOCK`
 ffconf.h, 63

`_FS_MINIMIZE`
 ffconf.h, 63

`_FS_READONLY`
 ffconf.h, 63

`_FS_REENTRANT`
 ffconf.h, 63

`_FS_RPATH`
 ffconf.h, 63

`_FS_TIMEOUT`
 ffconf.h, 63

`_FS_TINY`
 ffconf.h, 63

`_LFN_UNICODE`
 ffconf.h, 63

`_MAX_LFN`
 ffconf.h, 63

`_MAX_SS`
 ffconf.h, 63

`_MULTI_PARTITION`
 ffconf.h, 64

`_SYNC_t`
 ffconf.h, 64

`_T`
 ff.h, 56

`_TEXT`
 ff.h, 56

`_USE_ERASE`
 ffconf.h, 64

`_USE_FASTSEEK`
 ffconf.h, 64

`_USE_FORWARD`
 ffconf.h, 64

`_USE_LABEL`
 ffconf.h, 64

`_USE_LFN`
 ffconf.h, 64

`_USE_MKFS`
 ffconf.h, 64

`_USE_STRFUNC`
 ffconf.h, 64

`_VOLUMES`
 ffconf.h, 64

`_WORD_ACCESS`
 ffconf.h, 64

`ABORT`
 ff.c, 43

`ACMD13`
 sdmm.c, 67

`ACMD23`
 sdmm.c, 67

`ACMD41`
 sdmm.c, 67

`AM_ARC`
 ff.h, 56

`AM_DIR`
 ff.h, 56

`AM_HID`
 ff.h, 56

`AM_LFN`
 ff.h, 57

`AM_MASK`
 ff.h, 57

`AM_RDO`
 ff.h, 57

`AM_SYS`
 ff.h, 57

`AM_VOL`
 ff.h, 57

`ATA_GET_MODEL`
 diskio.h, 37

`ATA_GET_REV`
 diskio.h, 37

`ATA_GET_SN`
 diskio.h, 37

`BPB_BkBootSec`
 ff.c, 43

`BPB_BytsPerSec`

- ff.c, [43](#)
- BPB_ExtFlags
 - ff.c, [43](#)
- BPB_FATSz16
 - ff.c, [43](#)
- BPB_FATSz32
 - ff.c, [43](#)
- BPB_FSInfo
 - ff.c, [43](#)
- BPB_FSVer
 - ff.c, [43](#)
- BPB_HiddSec
 - ff.c, [43](#)
- BPB_Media
 - ff.c, [43](#)
- BPB_NumFATs
 - ff.c, [43](#)
- BPB_NumHeads
 - ff.c, [43](#)
- BPB_RootClus
 - ff.c, [44](#)
- BPB_RootEntCnt
 - ff.c, [44](#)
- BPB_RsvdSecCnt
 - ff.c, [44](#)
- BPB_SecPerClus
 - ff.c, [44](#)
- BPB_SecPerTrk
 - ff.c, [44](#)
- BPB_TotSec16
 - ff.c, [44](#)
- BPB_TotSec32
 - ff.c, [44](#)
- BS_55AA
 - ff.c, [44](#)
- BS_BootSig
 - ff.c, [44](#)
- BS_BootSig32
 - ff.c, [44](#)
- BS_DrvNum
 - ff.c, [44](#)
- BS_DrvNum32
 - ff.c, [44](#)
- BS_FilSysType
 - ff.c, [44](#)
- BS_FilSysType32
 - ff.c, [44](#)
- BS_OEMName
 - ff.c, [45](#)
- BS_VolID
 - ff.c, [45](#)
- BS_VolID32
 - ff.c, [45](#)
- BS_VolLab
 - ff.c, [45](#)
- BS_VolLab32
 - ff.c, [45](#)
- BS_jmpBoot
 - ff.c, [44](#)
- BYTE
 - integer.h, [65](#)
- bank1_note
 - order, [12](#)
- bank1_startstop
 - order, [12](#)
- bank2_note
 - order, [12](#)
- bank2_startstop
 - order, [12](#)
- bank3_note
 - order, [13](#)
- bank3_startstop
 - order, [13](#)
- bit_count
 - keyboard.c, [18](#)
- CHAR
 - integer.h, [65](#)
- CK_DQ
 - sdmm.c, [67](#)
- CK_H
 - sdmm.c, [67](#)
- CK_INIT
 - sdmm.c, [67](#)
- CK_L
 - sdmm.c, [68](#)
- CMD0
 - sdmm.c, [68](#)
- CMD1
 - sdmm.c, [68](#)
- CMD10
 - sdmm.c, [68](#)
- CMD12
 - sdmm.c, [68](#)
- CMD13
 - sdmm.c, [68](#)
- CMD16
 - sdmm.c, [68](#)
- CMD17
 - sdmm.c, [68](#)
- CMD18
 - sdmm.c, [68](#)
- CMD23
 - sdmm.c, [68](#)
- CMD24
 - sdmm.c, [68](#)
- CMD25
 - sdmm.c, [68](#)

CMD32
 sdmm.c, 68

CMD33
 sdmm.c, 69

CMD38
 sdmm.c, 69

CMD55
 sdmm.c, 69

CMD58
 sdmm.c, 69

CMD8
 sdmm.c, 69

CMD9
 sdmm.c, 69

COMM_PORT
 lcd8bit.c, 20

CREATE_LINKMAP
 ff.h, 57

CS_DQ
 sdmm.c, 69

CS_H
 sdmm.c, 69

CS_INIT
 sdmm.c, 69

CS_L
 sdmm.c, 69

CT_BLOCK
 diskio.h, 37

CT_MMC
 diskio.h, 37

CT_SD1
 diskio.h, 37

CT_SD2
 diskio.h, 37

CT_SDC
 diskio.h, 37

CTRL_ERASE_SECTOR
 diskio.h, 37

CTRL_POWER
 diskio.h, 37

CTRL_SYNC
 diskio.h, 37

caps_lock
 keyboard.c, 18

CardType
 sdmm.c, 72

char_waiting
 keyboard.c, 18

check_fs
 ff.c, 49

chk_chr
 ff.c, 49

chk_mounted
 ff.c, 49

clock_init
 keyboard.c, 17

clust
 DIR, 5
 FIL, 9

clust2sect
 ff.c, 49

config.h, 15
 F_CPU, 15

create_chain
 ff.c, 49

create_name
 ff.c, 49

csize
 FATFS, 7

currentVoice
 wavegen.c, 75

DATA_PORT
 lcd8bit.c, 20

DDE
 ff.c, 45

DEF_NAMEBUF
 ff.c, 45

DEVICE
 Makefile, 35

DI_DQ
 sdmm.c, 69

DI_H
 sdmm.c, 69

DI_INIT
 sdmm.c, 69

DI_L
 sdmm.c, 69

DIR, 5
 clust, 5
 dir, 6
 fn, 6
 fs, 6
 id, 6
 index, 6
 sclust, 6
 sect, 6

DIR_Attr
 ff.c, 45

DIR_CrtDate
 ff.c, 45

DIR_CrtTime
 ff.c, 45

DIR_CrtTimeTenth
 ff.c, 45

DIR_FileSize
 ff.c, 45

DIR_FstClusHI

- ff.c, 45
- DIR_FstClusLO
 - ff.c, 45
- DIR_LstAccDate
 - ff.c, 45
- DIR_NTres
 - ff.c, 46
- DIR_Name
 - ff.c, 46
- DIR_WrtDate
 - ff.c, 46
- DIR_WrtTime
 - ff.c, 46
- DO
 - sdmm.c, 70
- DO_DQ
 - sdmm.c, 70
- DO_INIT
 - sdmm.c, 70
- DRESULT
 - diskio.h, 38
- DSTATUS
 - diskio.h, 38
- DWORD
 - integer.h, 65
- data
 - onode, 12
- database
 - FATFS, 7
- deleteJustList
 - list.c, 25
 - list.h, 30
- deleteList
 - list.c, 25
 - list.h, 30
- deleteNode
 - list.c, 26
 - list.h, 30
- deleteNodeOnly
 - list.c, 26
 - list.h, 30
- deselect
 - sdmm.c, 70
- die
 - main.c, 34
- dir
 - DIR, 6
- dir_alloc
 - ff.c, 49
- dir_find
 - ff.c, 49
- dir_next
 - ff.c, 50
- dir_ptr
 - FIL, 9
- dir_read
 - ff.c, 50
- dir_register
 - ff.c, 50
- dir_sdi
 - ff.c, 50
- dir_sect
 - FIL, 10
- dirbase
 - FATFS, 7
- disk_initialize
 - diskio.h, 38
 - sdmm.c, 70
- disk_ioctl
 - diskio.h, 39
 - sdmm.c, 70
- disk_read
 - diskio.h, 39
 - sdmm.c, 70
- disk_status
 - diskio.h, 39
 - sdmm.c, 70
- disk_write
 - diskio.h, 39
 - sdmm.c, 71
- diskio.h
 - RES_ERROR, 38
 - RES_NOTRDY, 38
 - RES_OK, 38
 - RES_PARERR, 38
 - RES_WRPRT, 38
- diskio.h
 - ATA_GET_MODEL, 37
 - ATA_GET_REV, 37
 - ATA_GET_SN, 37
 - CT_BLOCK, 37
 - CT_MMC, 37
 - CT_SD1, 37
 - CT_SD2, 37
 - CT_SDC, 37
 - CTRL_ERASE_SECTOR, 37
 - CTRL_POWER, 37
 - CTRL_SYNC, 37
 - DRESULT, 38
 - DSTATUS, 38
 - disk_initialize, 38
 - disk_ioctl, 39
 - disk_read, 39
 - disk_status, 39
 - disk_write, 39
 - GET_BLOCK_SIZE, 37
 - GET_SECTOR_COUNT, 38
 - MMC_GET_CID, 38

- MMC_GET_CSD, [38](#)
- MMC_GET_OCR, [38](#)
- MMC_GET_SDSTAT, [38](#)
- MMC_GET_TYPE, [38](#)
- STA_NODISK, [38](#)
- STA_NOINIT, [38](#)
- STA_PROTECT, [38](#)
- dly_us
 - sdmm.c, [71](#)
- drv
 - FATFS, [7](#)
- dsect
 - FIL, [10](#)
- ENTER_FF
 - ff.c, [46](#)
- EOF
 - ff.h, [57](#)
- evictNode
 - list.c, [26](#)
 - list.h, [31](#)
- ExCvt
 - ff.c, [53](#)
- extended
 - keyboard.c, [18](#)
- FR_DENIED
 - ff.h, [59](#)
- FR_DISK_ERR
 - ff.h, [59](#)
- FR_EXIST
 - ff.h, [59](#)
- FR_INT_ERR
 - ff.h, [59](#)
- FR_INVALID_DRIVE
 - ff.h, [59](#)
- FR_INVALID_NAME
 - ff.h, [59](#)
- FR_INVALID_OBJECT
 - ff.h, [59](#)
- FR_INVALID_PARAMETER
 - ff.h, [59](#)
- FR_LOCKED
 - ff.h, [59](#)
- FR_MKFS_ABORTED
 - ff.h, [59](#)
- FR_NO_FILE
 - ff.h, [59](#)
- FR_NO_FILESYSTEM
 - ff.h, [59](#)
- FR_NO_PATH
 - ff.h, [59](#)
- FR_NOT_ENABLED
 - ff.h, [59](#)
- FR_NOT_ENOUGH_CORE
 - ff.h, [59](#)
- FR_NOT_READY
 - ff.h, [59](#)
- FR_OK
 - ff.h, [59](#)
- FR_TIMEOUT
 - ff.h, [59](#)
- FR_TOO_MANY_OPEN_FILES
 - ff.h, [59](#)
- FR_WRITE_PROTECTED
 - ff.h, [59](#)
- F_CPU
 - config.h, [15](#)
 - timeout.h, [72](#)
- f_chdir
 - ff.h, [60](#)
- f_chdrive
 - ff.h, [60](#)
- f_chmod
 - ff.h, [60](#)
- f_close
 - ff.c, [50](#)
 - ff.h, [60](#)
- f_eof
 - ff.h, [57](#)
- f_error
 - ff.h, [57](#)
- f_fdisk
 - ff.h, [60](#)
- f_forward
 - ff.h, [60](#)
- f_getcwd
 - ff.h, [60](#)
- f_getfree
 - ff.h, [60](#)
- f_getlabel
 - ff.c, [50](#)
 - ff.h, [60](#)
- f_gets
 - ff.h, [60](#)
- f_lseek
 - ff.c, [50](#)
 - ff.h, [60](#)
- f_mkdir
 - ff.h, [60](#)
- f_mkfs
 - ff.h, [60](#)
- f_mount
 - ff.c, [51](#)
 - ff.h, [60](#)
- f_open
 - ff.c, [51](#)
 - ff.h, [60](#)
- f_opendir

- ff.h, 61
- f_printf
 - ff.h, 61
- f_putc
 - ff.h, 61
- f_puts
 - ff.h, 61
- f_read
 - ff.c, 51
 - ff.h, 61
- f_readdir
 - ff.h, 61
- f_rename
 - ff.h, 61
- f_setlabel
 - ff.c, 51
 - ff.h, 61
- f_size
 - ff.h, 57
- f_stat
 - ff.h, 61
- f_sync
 - ff.c, 51
 - ff.h, 61
- f_tell
 - ff.h, 57
- f_truncate
 - ff.h, 61
- f_unlink
 - ff.h, 61
- f_utime
 - ff.h, 62
- f_write
 - ff.c, 51
 - ff.h, 62
- FA__DIRTY
 - ff.h, 57
- FA__ERROR
 - ff.h, 57
- FA__WRITTEN
 - ff.h, 57
- FA_CREATE_ALWAYS
 - ff.h, 57
- FA_CREATE_NEW
 - ff.h, 58
- FA_OPEN_ALWAYS
 - ff.h, 58
- FA_OPEN_EXISTING
 - ff.h, 58
- FA_READ
 - ff.h, 58
- FA_WRITE
 - ff.h, 58
- FATFS, 6
- csize, 7
- database, 7
- dirbase, 7
- drv, 7
- fatbase, 7
- free_clust, 7
- fs_type, 7
- fsi_flag, 7
- fsi_sector, 8
- fsize, 8
- id, 8
- last_clust, 8
- n_fatent, 8
- n_fats, 8
- n_rootdir, 8
- volbase, 8
- wflag, 8
- win, 8
- winsect, 8
- FIL, 9
 - clust, 9
 - dir_ptr, 9
 - dir_sect, 10
 - dsect, 10
 - flag, 10
 - fptr, 10
 - fs, 10
 - fsize, 10
 - id, 10
 - pad1, 10
 - sclust, 10
- FILINFO, 10
 - fattrib, 11
 - fdate, 11
 - fname, 11
 - fsize, 11
 - ftime, 11
- FINT
 - wavegen.h, 77
- FREE_BUF
 - ff.c, 46
- FRESULT
 - ff.h, 59
- FS
 - wavegen.h, 77
- FS_FAT12
 - ff.h, 58
- FS_FAT16
 - ff.h, 58
- FS_FAT32
 - ff.h, 58
- FSI_Free_Count
 - ff.c, 46
- FSI_LeadSig

- ff.c, [46](#)
- FSI_Nxt_Free
 - ff.c, [46](#)
- FSI_StrucSig
 - ff.c, [46](#)
- FatFs
 - ff.c, [53](#)
 - main.c, [34](#)
- fatbase
 - FATFS, [7](#)
- fattrib
 - FILINFO, [11](#)
- fdate
 - FILINFO, [11](#)
- ff.h
 - FR_DENIED, [59](#)
 - FR_DISK_ERR, [59](#)
 - FR_EXIST, [59](#)
 - FR_INT_ERR, [59](#)
 - FR_INVALID_DRIVE, [59](#)
 - FR_INVALID_NAME, [59](#)
 - FR_INVALID_OBJECT, [59](#)
 - FR_INVALID_PARAMETER, [59](#)
 - FR_LOCKED, [59](#)
 - FR_MKFS_ABORTED, [59](#)
 - FR_NO_FILE, [59](#)
 - FR_NO_FILESYSTEM, [59](#)
 - FR_NO_PATH, [59](#)
 - FR_NOT_ENABLED, [59](#)
 - FR_NOT_ENOUGH_CORE, [59](#)
 - FR_NOT_READY, [59](#)
 - FR_OK, [59](#)
 - FR_TIMEOUT, [59](#)
 - FR_TOO_MANY_OPEN_FILES, [59](#)
 - FR_WRITE_PROTECTED, [59](#)
- ff.c
 - _DF1S, [42](#)
 - _EXCVT, [42](#)
 - ABORT, [43](#)
 - BPB_BkBootSec, [43](#)
 - BPB_BytsPerSec, [43](#)
 - BPB_ExtFlags, [43](#)
 - BPB_FATSz16, [43](#)
 - BPB_FATSz32, [43](#)
 - BPB_FSInfo, [43](#)
 - BPB_FSVer, [43](#)
 - BPB_HiddSec, [43](#)
 - BPB_Media, [43](#)
 - BPB_NumFATs, [43](#)
 - BPB_NumHeads, [43](#)
 - BPB_RootClus, [44](#)
 - BPB_RootEntCnt, [44](#)
 - BPB_RsvdSecCnt, [44](#)
 - BPB_SecPerClus, [44](#)
 - BPB_SecPerTrk, [44](#)
 - BPB_TotSec16, [44](#)
 - BPB_TotSec32, [44](#)
 - BS_55AA, [44](#)
 - BS_BootSig, [44](#)
 - BS_BootSig32, [44](#)
 - BS_DrvNum, [44](#)
 - BS_DrvNum32, [44](#)
 - BS_FilSysType, [44](#)
 - BS_FilSysType32, [44](#)
 - BS_OEMName, [45](#)
 - BS_VolID, [45](#)
 - BS_VolID32, [45](#)
 - BS_VolLab, [45](#)
 - BS_VolLab32, [45](#)
 - BS_jmpBoot, [44](#)
 - check_fs, [49](#)
 - chk_chr, [49](#)
 - chk_mounted, [49](#)
 - clust2sect, [49](#)
 - create_chain, [49](#)
 - create_name, [49](#)
 - DDE, [45](#)
 - DEF_NAMEBUF, [45](#)
 - DIR_Attr, [45](#)
 - DIR_CrtDate, [45](#)
 - DIR_CrtTime, [45](#)
 - DIR_CrtTimeTenth, [45](#)
 - DIR_FileSize, [45](#)
 - DIR_FstClusHI, [45](#)
 - DIR_FstClusLO, [45](#)
 - DIR_LstAccDate, [45](#)
 - DIR_NTres, [46](#)
 - DIR_Name, [46](#)
 - DIR_WrtDate, [46](#)
 - DIR_WrtTime, [46](#)
 - dir_alloc, [49](#)
 - dir_find, [49](#)
 - dir_next, [50](#)
 - dir_read, [50](#)
 - dir_register, [50](#)
 - dir_sdi, [50](#)
 - ENTER_FF, [46](#)
 - ExCvt, [53](#)
 - f_close, [50](#)
 - f_getlabel, [50](#)
 - f_lseek, [50](#)
 - f_mount, [51](#)
 - f_open, [51](#)
 - f_read, [51](#)
 - f_setlabel, [51](#)
 - f_sync, [51](#)
 - f_write, [51](#)
 - FREE_BUF, [46](#)

- FSI_Free_Count, [46](#)
- FSI_LeadSig, [46](#)
- FSI_Nxt_Free, [46](#)
- FSI_StrucSig, [46](#)
- FatFs, [53](#)
- follow_path, [52](#)
- Fsid, [53](#)
- get_fat, [52](#)
- INIT_BUF, [46](#)
- IsDBCS1, [46](#)
- IsDBCS2, [47](#)
- IsDigit, [47](#)
- IsLower, [47](#)
- IsUpper, [47](#)
- LDIR_Attr, [47](#)
- LDIR_Chksum, [47](#)
- LDIR_FstClusLO, [47](#)
- LDIR_Ord, [47](#)
- LDIR_Type, [47](#)
- LEAVE_FF, [47](#)
- LLE, [47](#)
- ld_clust, [52](#)
- MBR_Table, [47](#)
- MIN_FAT16, [47](#)
- MIN_FAT32, [47](#)
- mem_cmp, [52](#)
- mem_cpy, [52](#)
- mem_set, [52](#)
- move_window, [52](#)
- NDDE, [47](#)
- NS, [48](#)
- NS_BODY, [48](#)
- NS_DOT, [48](#)
- NS_EXT, [48](#)
- NS_LAST, [48](#)
- NS_LFN, [48](#)
- NS_LOSS, [48](#)
- put_fat, [52](#)
- remove_chain, [52](#)
- SS, [48](#)
- SZ_DIR, [48](#)
- SZ_PTE, [48](#)
- st_clust, [53](#)
- sync_fs, [53](#)
- sync_window, [53](#)
- validate, [53](#)
- ff.h
 - _FATFS, [56](#)
 - _T, [56](#)
 - _TEXT, [56](#)
 - AM_ARC, [56](#)
 - AM_DIR, [56](#)
 - AM_HID, [56](#)
 - AM_LFN, [57](#)
 - AM_MASK, [57](#)
 - AM_RDO, [57](#)
 - AM_SYS, [57](#)
 - AM_VOL, [57](#)
 - CREATE_LINKMAP, [57](#)
 - EOF, [57](#)
 - f_chdir, [60](#)
 - f_chdrive, [60](#)
 - f_chmod, [60](#)
 - f_close, [60](#)
 - f_eof, [57](#)
 - f_error, [57](#)
 - f_fdisk, [60](#)
 - f_forward, [60](#)
 - f_getcwd, [60](#)
 - f_getfree, [60](#)
 - f_getlabel, [60](#)
 - f_gets, [60](#)
 - f_lseek, [60](#)
 - f_mkdir, [60](#)
 - f_mkfs, [60](#)
 - f_mount, [60](#)
 - f_open, [60](#)
 - f_opendir, [61](#)
 - f_printf, [61](#)
 - f_putc, [61](#)
 - f_puts, [61](#)
 - f_read, [61](#)
 - f_readdir, [61](#)
 - f_rename, [61](#)
 - f_setlabel, [61](#)
 - f_size, [57](#)
 - f_stat, [61](#)
 - f_sync, [61](#)
 - f_tell, [57](#)
 - f_truncate, [61](#)
 - f_unlink, [61](#)
 - f_utime, [62](#)
 - f_write, [62](#)
 - FA_DIRTY, [57](#)
 - FA_ERROR, [57](#)
 - FA_WRITTEN, [57](#)
 - FA_CREATE_ALWAYS, [57](#)
 - FA_CREATE_NEW, [58](#)
 - FA_OPEN_ALWAYS, [58](#)
 - FA_OPEN_EXISTING, [58](#)
 - FA_READ, [58](#)
 - FA_WRITE, [58](#)
 - FRESULT, [59](#)
 - FS_FAT12, [58](#)
 - FS_FAT16, [58](#)
 - FS_FAT32, [58](#)
 - get_fattime, [62](#)
 - LD2PD, [58](#)

- LD2PT, [58](#)
- LD_DWORD, [58](#)
- LD_WORD, [58](#)
- ST_DWORD, [59](#)
- ST_WORD, [59](#)
- TCHAR, [59](#)
- ffconf.h
 - _CODE_PAGE, [63](#)
 - _FFCONF, [63](#)
 - _FS_LOCK, [63](#)
 - _FS_MINIMIZE, [63](#)
 - _FS_READONLY, [63](#)
 - _FS_REENTRANT, [63](#)
 - _FS_RPATH, [63](#)
 - _FS_TIMEOUT, [63](#)
 - _FS_TINY, [63](#)
 - _LFN_UNICODE, [63](#)
 - _MAX_LFN, [63](#)
 - _MAX_SS, [63](#)
 - _MULTI_PARTITION, [64](#)
 - _SYNC_t, [64](#)
 - _USE_ERASE, [64](#)
 - _USE_FASTSEEK, [64](#)
 - _USE_FORWARD, [64](#)
 - _USE_LABEL, [64](#)
 - _USE_LFN, [64](#)
 - _USE_MKFS, [64](#)
 - _USE_STRFUNC, [64](#)
 - _VOLUMES, [64](#)
 - _WORD_ACCESS, [64](#)
- flag
 - FIL, [10](#)
- fn
 - DIR, [6](#)
- fname
 - FILINFO, [11](#)
- follow_path
 - ff.c, [52](#)
- fp
 - main.c, [34](#)
- fptr
 - FIL, [10](#)
- free_clust
 - FATFS, [7](#)
- frequencyCoef
 - wavegen.c, [75](#)
- frequencyCoef2
 - wavegen.c, [75](#)
- fs
 - DIR, [6](#)
 - FIL, [10](#)
- fs_type
 - FATFS, [7](#)
- fsi_flag
 - FATFS, [7](#)
- fsi_sector
 - FATFS, [8](#)
- Fsid
 - ff.c, [53](#)
- fsize
 - FATFS, [8](#)
 - FIL, [10](#)
 - FILINFO, [11](#)
- ftime
 - FILINFO, [11](#)
- g
 - list.c, [28](#)
- GET_BLOCK_SIZE
 - diskio.h, [37](#)
- GET_SECTOR_COUNT
 - diskio.h, [38](#)
- get_fat
 - ff.c, [52](#)
- get_fattime
 - ff.h, [62](#)
 - main.c, [34](#)
- getNextOrder
 - list.c, [26](#)
 - list.h, [31](#)
- getOrderData
 - list.c, [26](#)
 - list.h, [31](#)
- getOrderId
 - list.c, [26](#)
 - list.h, [31](#)
- getOrderNode
 - list.c, [26](#)
 - list.h, [31](#)
- getOrderNote
 - list.c, [27](#)
 - list.h, [31](#)
- getOrderStartstop
 - list.c, [27](#)
 - list.h, [31](#)
- getPrevOrder
 - list.c, [27](#)
 - list.h, [32](#)
- INIT_BUF
 - ff.c, [46](#)
- INT
 - integer.h, [65](#)
- INTERRUPT_PERIOD
 - wavegen.h, [77](#)
- ISR
 - keyboard.c, [17](#)
 - wavegen.c, [74](#)
- id

- DIR, [6](#)
- FATFS, [8](#)
- FIL, [10](#)
- order, [13](#)
- index
 - DIR, [6](#)
- init_keyboard
 - keyboard.c, [17](#)
- insertNode
 - list.c, [27](#)
 - list.h, [32](#)
- integer.h
 - BYTE, [65](#)
 - CHAR, [65](#)
 - DWORD, [65](#)
 - INT, [65](#)
 - LONG, [65](#)
 - SHORT, [65](#)
 - UCHAR, [65](#)
 - UINT, [65](#)
 - ULONG, [65](#)
 - USHORT, [65](#)
 - WCHAR, [65](#)
 - WORD, [65](#)
- IsDBCS1
 - ff.c, [46](#)
- IsDBCS2
 - ff.c, [47](#)
- IsDigit
 - ff.c, [47](#)
- IsLower
 - ff.c, [47](#)
- IsUpper
 - ff.c, [47](#)
- KB_CLK
 - keyboard.c, [17](#)
- KB_DATA
 - keyboard.c, [17](#)
- KB_PORT
 - keyboard.c, [17](#)
- kbd_data
 - keyboard.c, [18](#)
- keyboard.c
 - bit_count, [18](#)
 - caps_lock, [18](#)
 - char_waiting, [18](#)
 - clock_init, [17](#)
 - extended, [18](#)
 - ISR, [17](#)
 - init_keyboard, [17](#)
 - KB_CLK, [17](#)
 - KB_DATA, [17](#)
 - KB_PORT, [17](#)
 - kbd_data, [18](#)
 - main, [17](#)
 - read_char, [17](#)
 - release, [18](#)
 - render_scan_code, [17](#)
 - shift, [18](#)
 - st, [18](#)
 - started, [18](#)
- keyboard/keyboard.c, [16](#)
- keyboard/keymap.h, [19](#)
- keymap.h
 - PROGMEM, [19](#)
- LCD_E
 - lcd8bit.c, [20](#)
- LCD_RS
 - lcd8bit.c, [20](#)
- LD2PD
 - ff.h, [58](#)
- LD2PT
 - ff.h, [58](#)
- LD_DWORD
 - ff.h, [58](#)
- LD_WORD
 - ff.h, [58](#)
- LDIR_Attr
 - ff.c, [47](#)
- LDIR_Chksum
 - ff.c, [47](#)
- LDIR_FstClusLO
 - ff.c, [47](#)
- LDIR_Ord
 - ff.c, [47](#)
- LDIR_Type
 - ff.c, [47](#)
- LEAVE_FF
 - ff.c, [47](#)
- LLE
 - ff.c, [47](#)
- LONG
 - integer.h, [65](#)
- last_clust
 - FATFS, [8](#)
- lcd8bit.c
 - COMM_PORT, [20](#)
 - DATA_PORT, [20](#)
 - LCD_E, [20](#)
 - LCD_RS, [20](#)
 - lcd_clear_and_home, [20](#)
 - lcd_goto, [20](#)
 - lcd_home, [21](#)
 - lcd_init, [21](#)
 - lcd_line_one, [21](#)
 - lcd_line_two, [21](#)

- lcd_set_write_data, [21](#)
- lcd_set_write_instruction, [21](#)
- lcd_write_byte, [21](#)
- lcd_write_data, [21](#)
- lcd_write_string, [21](#)
- lcd_write_string_0, [22](#)
- lcd_write_string_p, [22](#)
- lcd8bit.h
 - lcd_clear_and_home, [23](#)
 - lcd_goto, [23](#)
 - lcd_home, [23](#)
 - lcd_init, [23](#)
 - lcd_line_one, [23](#)
 - lcd_line_two, [23](#)
 - lcd_set_write_data, [23](#)
 - lcd_set_write_instruction, [23](#)
 - lcd_write_byte, [23](#)
 - lcd_write_data, [24](#)
 - lcd_write_string, [24](#)
 - lcd_write_string_0, [24](#)
 - lcd_write_string_p, [24](#)
- lcd8bit/lcd8bit.c, [19](#)
- lcd8bit/lcd8bit.h, [22](#)
- lcd_clear_and_home
 - lcd8bit.c, [20](#)
 - lcd8bit.h, [23](#)
- lcd_goto
 - lcd8bit.c, [20](#)
 - lcd8bit.h, [23](#)
- lcd_home
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_init
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_line_one
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_line_two
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_set_write_data
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_set_write_instruction
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_write_byte
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [23](#)
- lcd_write_data
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [24](#)
- lcd_write_string
 - lcd8bit.c, [21](#)
 - lcd8bit.h, [24](#)
- lcd_write_string_0
 - lcd8bit.c, [22](#)
 - lcd8bit.h, [24](#)
- lcd_write_string_p
 - lcd8bit.c, [22](#)
 - lcd8bit.h, [24](#)
- ld_clust
 - ff.c, [52](#)
- list.c
 - deleteJustList, [25](#)
 - deleteList, [25](#)
 - deleteNode, [26](#)
 - deleteNodeOnly, [26](#)
 - evictNode, [26](#)
 - g, [28](#)
 - getNextOrder, [26](#)
 - getOrderData, [26](#)
 - getOrderId, [26](#)
 - getNodeOrder, [26](#)
 - getNodeOrder, [27](#)
 - getNodeStartstop, [27](#)
 - getNodePrevOrder, [27](#)
 - insertNode, [27](#)
 - newNode, [27](#)
 - newNodeByRef, [27](#)
 - printList, [27](#)
 - pushNode, [28](#)
 - setOrderId, [28](#)
 - setOrderNote, [28](#)
 - setOrderStartstop, [28](#)
 - sort, [28](#)
 - swap, [28](#)
- list.h
 - deleteJustList, [30](#)
 - deleteList, [30](#)
 - deleteNode, [30](#)
 - deleteNodeOnly, [30](#)
 - evictNode, [31](#)
 - getNextOrder, [31](#)
 - getOrderData, [31](#)
 - getOrderId, [31](#)
 - getNodeOrder, [31](#)
 - getNodeOrder, [31](#)
 - getNodeStartstop, [31](#)
 - getNodePrevOrder, [32](#)
 - insertNode, [32](#)
 - newNode, [32](#)
 - newNodeByRef, [32](#)
 - printList, [32](#)
 - pushNode, [32](#)
 - setOrderId, [33](#)
 - setOrderNote, [33](#)

- setOrderStartstop, 33
 - sort, 33
 - swap, 33
- list/list.c, 24
- list/list.h, 29
- MBR_Table
 - ff.c, 47
- MIN_FAT16
 - ff.c, 47
- MIN_FAT32
 - ff.c, 47
- MMC_GET_CID
 - diskio.h, 38
- MMC_GET_CSD
 - diskio.h, 38
- MMC_GET_OCR
 - diskio.h, 38
- MMC_GET_SDSTAT
 - diskio.h, 38
- MMC_GET_TYPE
 - diskio.h, 38
- main
 - keyboard.c, 17
 - main.c, 34
- main.c, 33
 - die, 34
 - FatFs, 34
 - fp, 34
 - get_fattime, 34
 - main, 34
- Makefile, 35
 - DEVICE, 35
 - WI, 35
- mem_cmp
 - ff.c, 52
- mem_cpy
 - ff.c, 52
- mem_set
 - ff.c, 52
- move_window
 - ff.c, 52
- n_fatent
 - FATFS, 8
- n_fats
 - FATFS, 8
- n_rootdir
 - FATFS, 8
- NDDE
 - ff.c, 47
- NS
 - ff.c, 48
- NS_BODY
 - ff.c, 48
- NS_DOT
 - ff.c, 48
- NS_EXT
 - ff.c, 48
- NS_LAST
 - ff.c, 48
- NS_LFN
 - ff.c, 48
- NS_LOSS
 - ff.c, 48
- newNode
 - list.c, 27
 - list.h, 32
- newNodeByRef
 - list.c, 27
 - list.h, 32
- next
 - onode, 12
- onode, 11
 - data, 12
 - next, 12
 - prev, 12
- order, 12
 - bank1_note, 12
 - bank1_startstop, 12
 - bank2_note, 12
 - bank2_startstop, 12
 - bank3_note, 13
 - bank3_startstop, 13
 - id, 13
- PROGMEM
 - keymap.h, 19
 - wavegen.c, 75
- pad1
 - FIL, 10
- prev
 - onode, 12
- printList
 - list.c, 27
 - list.h, 32
- pushNode
 - list.c, 28
 - list.h, 32
- put_fat
 - ff.c, 52
- RES_ERROR
 - diskio.h, 38
- RES_NOTRDY
 - diskio.h, 38
- RES_OK
 - diskio.h, 38
- RES_PARERR

- diskio.h, [38](#)
- RES_WRPRT
 - diskio.h, [38](#)
- rcvr_datablock
 - sdmm.c, [71](#)
- rcvr_mmc
 - sdmm.c, [71](#)
- read_char
 - keyboard.c, [17](#)
- release
 - keyboard.c, [18](#)
- remove_chain
 - ff.c, [52](#)
- render_scan_code
 - keyboard.c, [17](#)
- SD_COM_PORT
 - sdmm.c, [70](#)
- SHORT
 - integer.h, [65](#)
- SS
 - ff.c, [48](#)
- ST_DWORD
 - ff.h, [59](#)
- ST_WORD
 - ff.h, [59](#)
- STA_NODISK
 - diskio.h, [38](#)
- STA_NOINIT
 - diskio.h, [38](#)
- STA_PROTECT
 - diskio.h, [38](#)
- SZ_DIR
 - ff.c, [48](#)
- SZ_PTE
 - ff.c, [48](#)
- sclust
 - DIR, [6](#)
 - FIL, [10](#)
- sd_card/diskio.h, [35](#)
- sd_card/ff.c, [39](#)
- sd_card/ff.h, [54](#)
- sd_card/ffconf.h, [62](#)
- sd_card/integer.h, [64](#)
- sd_card/sdmm.c, [65](#)
- sd_card/timeout.h, [72](#)
- sdmm.c
 - ACMD13, [67](#)
 - ACMD23, [67](#)
 - ACMD41, [67](#)
 - CK_DQ, [67](#)
 - CK_H, [67](#)
 - CK_INIT, [67](#)
 - CK_L, [68](#)
- CMD0, [68](#)
- CMD1, [68](#)
- CMD10, [68](#)
- CMD12, [68](#)
- CMD13, [68](#)
- CMD16, [68](#)
- CMD17, [68](#)
- CMD18, [68](#)
- CMD23, [68](#)
- CMD24, [68](#)
- CMD25, [68](#)
- CMD32, [68](#)
- CMD33, [69](#)
- CMD38, [69](#)
- CMD55, [69](#)
- CMD58, [69](#)
- CMD8, [69](#)
- CMD9, [69](#)
- CS_DQ, [69](#)
- CS_H, [69](#)
- CS_INIT, [69](#)
- CS_L, [69](#)
- CardType, [72](#)
- DI_DQ, [69](#)
- DI_H, [69](#)
- DI_INIT, [69](#)
- DI_L, [69](#)
- DO, [70](#)
- DO_DQ, [70](#)
- DO_INIT, [70](#)
- deselect, [70](#)
- disk_initialize, [70](#)
- disk_ioctl, [70](#)
- disk_read, [70](#)
- disk_status, [70](#)
- disk_write, [71](#)
- dly_us, [71](#)
- rcvr_datablock, [71](#)
- rcvr_mmc, [71](#)
- SD_COM_PORT, [70](#)
- select, [71](#)
- send_cmd, [71](#)
- Stat, [72](#)
- wait_ready, [71](#)
- xmit_datablock, [71](#)
- xmit_mmc, [72](#)
- sect
 - DIR, [6](#)
- select
 - sdmm.c, [71](#)
- send_cmd
 - sdmm.c, [71](#)
- setOrderId
 - list.c, [28](#)

- list.h, 33
- setOrderNote
 - list.c, 28
 - list.h, 33
- setOrderStartstop
 - list.c, 28
 - list.h, 33
- shift
 - keyboard.c, 18
- sort
 - list.c, 28
 - list.h, 33
- sound1Enabled
 - wavegen.c, 75
- sound2Enabled
 - wavegen.c, 75
- sound3Enabled
 - wavegen.c, 75
- soundPWM
 - wavegen.c, 75
- st
 - keyboard.c, 18
- st_clust
 - ff.c, 53
- started
 - keyboard.c, 18
- Stat
 - sdmm.c, 72
- swap
 - list.c, 28
 - list.h, 33
- sync_fs
 - ff.c, 53
- sync_window
 - ff.c, 53
- TCHAR
 - ff.h, 59
- timeout.h
 - F_CPU, 72
- UCHAR
 - integer.h, 65
- UINT
 - integer.h, 65
- ULONG
 - integer.h, 65
- USHORT
 - integer.h, 65
- validate
 - ff.c, 53
- volbase
 - FATFS, 8

- WCHAR
 - integer.h, 65
- WORD
 - integer.h, 65
- wait_ready
 - sdmm.c, 71
- wavegen.c
 - currentVoice, 75
 - frequencyCoef, 75
 - frequencyCoef2, 75
 - ISR, 74
 - PROGMEM, 75
 - sound1Enabled, 75
 - sound2Enabled, 75
 - sound3Enabled, 75
 - soundPWM, 75
 - wavegen_clock_init, 74
 - wavegen_disableSound, 74
 - wavegen_noiseWave, 74
 - wavegen_pwmInit, 74
 - wavegen_pwmSet, 74
 - wavegen_setFrequency, 74
 - wavegen_setFrequency2, 74
 - wavegen_setSound, 74
 - wavegen_sineWave, 74
 - wavetable, 76
 - wavetable2, 76
 - wavetable3, 76
- wavegen.h
 - FINT, 77
 - FS, 77
 - INTERRUPT_PERIOD, 77
 - wavegen_clock_init, 77
 - wavegen_disableSound, 77
 - wavegen_noiseWave, 78
 - wavegen_pwmInit, 78
 - wavegen_pwmSet, 78
 - wavegen_setFrequency, 78
 - wavegen_setFrequency2, 78
 - wavegen_setSound, 78
 - wavegen_sineWave, 78
- wavegen/wavegen.c, 73
- wavegen/wavegen.h, 76
- wavegen_clock_init
 - wavegen.c, 74
 - wavegen.h, 77
- wavegen_disableSound
 - wavegen.c, 74
 - wavegen.h, 77
- wavegen_noiseWave
 - wavegen.c, 74
 - wavegen.h, 78
- wavegen_pwmInit
 - wavegen.c, 74

- wavegen.h, [78](#)
- wavegen_pwmSet
 - wavegen.c, [74](#)
 - wavegen.h, [78](#)
- wavegen_setFrequency
 - wavegen.c, [74](#)
 - wavegen.h, [78](#)
- wavegen_setFrequency2
 - wavegen.c, [74](#)
 - wavegen.h, [78](#)
- wavegen_setSound
 - wavegen.c, [74](#)
 - wavegen.h, [78](#)
- wavegen_sineWave
 - wavegen.c, [74](#)
 - wavegen.h, [78](#)
- wavetable
 - wavegen.c, [76](#)
- wavetable2
 - wavegen.c, [76](#)
- wavetable3
 - wavegen.c, [76](#)
- wflag
 - FATFS, [8](#)
- win
 - FATFS, [8](#)
- winsect
 - FATFS, [8](#)
- WI
 - Makefile, [35](#)
- xmit_datablock
 - sdmm.c, [71](#)
- xmit_mmc
 - sdmm.c, [72](#)