# EyeBot\_BB

Generated by Doxygen 1.7.6.1

Sat Sep 14 2013 08:41:43

# **Contents**

1	EyeE	Bot_BB Documentation	1
2	Clas	s Index	3
	2.1	Class List	3
3	File	index	5
	3.1	File List	5
4	Clas	s Documentation	7
	4.1	_HDTDevADC_t Struct Reference	7
	4.2	_HDTDevALL_t Struct Reference	8
	4.3	_HDTDevCAM_t Struct Reference	9
	4.4	_HDTDevCOM_t Struct Reference	9
	4.5	_HDTDevDRIVE_t Struct Reference	10
	4.6	_HDTDevENCODER_t Struct Reference	11
	4.7	_HDTDevice_t Struct Reference	12
	4.8	_HDTDevIRTV_t Struct Reference	13
	4.9	_HDTDevMOTOR_t Struct Reference	14
	4.10	_HDTDevPSD_t Struct Reference	15
	4.11	_HDTDevSERVO_t Struct Reference	15
	4.12	_HDTEntry_t Struct Reference	16
	4.13	_HDTTable_t Struct Reference	16
	4.14	_HDTTypes_t Struct Reference	17
	4.15	_process_t Struct Reference	17
		proclist_t Struct Reference	
	4.17	coord pair t Struct Reference	18

ii CONTENTS

		4.17.1	Detailed I	Description		18
	4.18	cursor_	_t Struct Re	eference		19
	4.19	fbinfo_t	t Struct Re	eference		19
	4.20	Hints S	truct Refe	rence		20
	4.21	info_cp	u_t Struct	Reference		20
	4.22	info_me	em_t Struc	ct Reference		20
	4.23	info_mi	isc_t Struc	t Reference		21
	4.24	info_pro	oc_t Struc	t Reference		21
	4.25	LCDHa	ındle Struc	ct Reference		21
		4.25.1	Detailed I	Description		23
	4.26	listmen	u_t Struct	Reference		23
	4.27	m6key_	_box_t Stru	uct Reference		24
	4.28	menuite	em_t Struc	ct Reference		24
	4.29	rect_t S	Struct Refe	erence		25
		4.29.1	Detailed I	Description		25
	4.30	screen	_t Struct R	Reference		25
	4.31	touch_e	event_t Str	ruct Reference		26
	4.32	touch_i	map_t Stru	uct Reference		27
_	Eile I	Daguma	ntetion			29
5	5.1		entation	de/adc.h File Reference		
	5.1	5.1.1				
				Description		
		5.1.2	5.1.2.1	Documentation		
			5.1.2.1	ConvADCSampleToVoltage		
			5.1.2.3	OSGetADC		
			5.1.2.4	OSInitADC		
	5.2	EveRot		de/camera.h File Reference		
	5.2	5.2.1		Description		
		5.2.2		Documentation		
		5.2.2	5.2.2.1	CAMGetFrameRGB		
	E O	EveDet	5.2.2.2	CAMInit		33
	5.3	_		de/eyebot.h File Reference		
		5.3.1	Detailed I	Description		34

CONTENTS iii

5.4	EyeBot	t_BB/include/globals.h File Reference
	5.4.1	Detailed Description
5.5	EyeBot	t_BB/include/hdt.h File Reference
	5.5.1	Detailed Description
	5.5.2	Function Documentation
		5.5.2.1 HDTClearADC
		5.5.2.2 HDTClearCAM
		5.5.2.3 HDTClearCOM
		5.5.2.4 HDTClearDRIVE
		5.5.2.5 HDTClearENCODER
		5.5.2.6 HDTClearIRTV
		5.5.2.7 HDTClearMOTOR
		5.5.2.8 HDTClearPSD
		5.5.2.9 HDTClearSERVO
		5.5.2.10 HDTClearTable
		5.5.2.11 HDTFindEntry
		5.5.2.12 HDTFindTable
		5.5.2.13 HDTLinkDRV2ENC
		5.5.2.14 HDTLinkENC2MOT
		5.5.2.15 HDTListEntry
		5.5.2.16 HDTLoadADC
		5.5.2.17 HDTLoadCAM
		5.5.2.18 HDTLoadCOM
		5.5.2.19 HDTLoadDRIVE
		5.5.2.20 HDTLoadENCODER
		5.5.2.21 HDTLoadIRTV
		5.5.2.22 HDTLoadMOTOR
		5.5.2.23 HDTLoadPSD
		5.5.2.24 HDTLoadSERVO
		5.5.2.25 HDTLoadTable
		5.5.2.26 HDTValidate
5.6	EyeBot	t_BB/include/irtv.h File Reference
	5.6.1	Detailed Description
	5.6.2	Function Documentation

iv CONTENTS

		5.6.2.1	IRTVInit
		5.6.2.2	IRTVRead 50
5.7	EyeBo	t_BB/inclu	de/key.h File Reference
	5.7.1	Detailed	Description
	5.7.2	Function	Documentation
		5.7.2.1	KEYDecode
		5.7.2.2	KEYGet
		5.7.2.3	KEYGetRAW
		5.7.2.4	KEYGetRegion
		5.7.2.5	KEYGetTM
		5.7.2.6	KEYGetXY
		5.7.2.7	KEYIdle
		5.7.2.8	KEYInit
		5.7.2.9	KEYNoTouch
		5.7.2.10	KEYRead 55
		5.7.2.11	KEYRelease
		5.7.2.12	KEYSetRegion
		5.7.2.13	KEYSetTM
		5.7.2.14	KEYWait
5.8	EyeBo	t_BB/inclu	de/latches.h File Reference
	5.8.1	Detailed	Description
	5.8.2	Function	Documentation
		5.8.2.1	OSLatchBankSetup
		5.8.2.2	OSLatchCleanup
		5.8.2.3	OSLatchInit
		5.8.2.4	OSLatchRead
		5.8.2.5	OSLatchSetup 60
		5.8.2.6	OSLatchWrite 60
5.9	EyeBo	t_BB/inclu	de/lcd.h File Reference 61
	5.9.1	Detailed	Description
	5.9.2	Function	Documentation
		5.9.2.1	getColor
		5.9.2.2	InvertColor
		5.9.2.3	LCDArea

CONTENTS

5.9.2.4	LCDAreaInvert
5.9.2.5	LCDClear 67
5.9.2.6	LCDFrame
5.9.2.7	LCDGetFBInfo 67
5.9.2.8	LCDGetList
5.9.2.9	LCDGetMode
5.9.2.10	LCDGetPixel
5.9.2.11	LCDGetPos
5.9.2.12	LCDInit
5.9.2.13	LCDInvertPixel 69
5.9.2.14	LCDLine
5.9.2.15	LCDLineInvert 69
5.9.2.16	LCDList
5.9.2.17	LCDListActiveItem
5.9.2.18	LCDListBox
5.9.2.19	LCDListCount
5.9.2.20	LCDListIndex
5.9.2.21	LCDListScrollDown
5.9.2.22	LCDListScrollUp
5.9.2.23	LCDMenu
5.9.2.24	LCDMenul
5.9.2.25	LCDMenuItem
5.9.2.26	LCDNeedRefresh
5.9.2.27	LCDPrintf
5.9.2.28	LCDPutChar
5.9.2.29	LCDPutFloat
5.9.2.30	LCDPutFloatS
5.9.2.31	LCDPutHex
5.9.2.32	LCDPutHex1
5.9.2.33	LCDPutImageRGB
5.9.2.34	LCDPutInt
5.9.2.35	LCDPutIntS
5.9.2.36	LCDPutString
5.9.2.37	LCDRefresh

vi CONTENTS

	5.9.2.38 LCDRelease	6
	5.9.2.39 LCDResetMode	6
	5.9.2.40 LCDSetChar	6
	5.9.2.41 LCDSetList	6
	5.9.2.42 LCDSetMode	7
	5.9.2.43 LCDSetPixel	7
	5.9.2.44 LCDSetPos	7
	5.9.2.45 LCDSetPrintf	8
	5.9.2.46 LCDSetString	8
	5.9.2.47 LCDTextBar	8
	5.9.2.48 LCDTextColor	9
5.10 EyeBo	ot_BB/include/misc.h File Reference	0
5.10.1	1 Detailed Description	1
5.11 EyeBo	ot_BB/include/multitasking.h File Reference	1
5.11.1	1 Detailed Description	1
5.12 EyeBo	ot_BB/include/params.h File Reference	2
5.12.1	1 Detailed Description	3
5.13 EyeBo	ot_BB/include/psd_sensors.h File Reference	3
5.13.1	1 Detailed Description	4
5.13.2	2 Function Documentation	4
	5.13.2.1 PSDGet	4
	5.13.2.2 PSDGetRaw	4
	5.13.2.3 PSDInit	5
	5.13.2.4 PSDRelease	5
5.14 EyeBo	ot_BB/include/servos_and_motors.h File Reference 8	5
5.14.1	1 Detailed Description	7
5.14.2	2 Function Documentation	7
	5.14.2.1 MOTORDrive	7
	5.14.2.2 MOTORInit	7
	5.14.2.3 MOTORRelease	8
	5.14.2.4 QUADRead	8
	5.14.2.5 QUADRelease	8
	5.14.2.6 QUADReset	9
	5.14.2.7 SERVOInit	9

CONTENTS vii

		5.14.2.8	SERVORelease							89
		5.14.2.9	SERVOSet							89
5.15	EyeBot	_BB/includ	e/spi.h File Reference							90
!	5.15.1	Detailed I	escription							91
!	5.15.2	Function	Occumentation							91
		5.15.2.1	SPIInit							91
		5.15.2.2	SPIRead							91
		5.15.2.3	SPIReadDefault							92
		5.15.2.4	SPIRelease							92
		5.15.2.5	SPISend							92
		5.15.2.6	SPISendDefault							92
5.16	EyeBot	_BB/includ	e/spi_commands.h File Reference	ce .						93
!	5.16.1	Detailed I	Description							94
5.17	EyeBot	_BB/includ	e/system.h File Reference							94
!	5.17.1	Detailed I	escription							95
!	5.17.2	Function	Documentation							95
		5.17.2.1	OSError							95
		5.17.2.2	OSInfoCPU							96
		5.17.2.3	OSInfoMem							96
		5.17.2.4	OSInfoMisc							96
		5.17.2.5	OSInfoProc							96
		5.17.2.6	OSMachineID							97
		5.17.2.7	OSMachineName							97
		5.17.2.8	OSMachineSpeed							97
		5.17.2.9	OSMachineType							97
		5.17.2.10	OSVersion							98
5.18	EyeBot	_BB/includ	e/timer.h File Reference							98
!	5.18.1	Detailed I	Description			٠				99
!	5.18.2	Function	Documentation							99
		5.18.2.1	OSWait							99
5.19	EyeBot	_BB/includ	e/types.h File Reference						. 1	100
			Description							
5.20	EyeBot	_BB/includ	e/vomega.h File Reference						. 1	102
!	5.20.1	Detailed I	escription						. 1	103

viii CONTENTS

5.20.2	2 Function Documentation
	5.20.2.1 VWDriveStraight
	5.20.2.2 VWDriveTurn
	5.20.2.3 VWDriveWait
	5.20.2.4 VWInit
5.21 EyeB	ot_BB/src/adc.c File Reference
5.21.	Detailed Description
5.21.2	2 Function Documentation
	5.21.2.1 ConvADCSampleToVoltage
	5.21.2.2 OSADCRelease
	5.21.2.3 OSGetADC
	5.21.2.4 OSInitADC
5.22 EyeB	ot_BB/src/camera.c File Reference
5.22.	Detailed Description
5.22.2	2 Function Documentation
	5.22.2.1 CAMGetFrameRGB
	5.22.2.2 CAMInit
5.23 EyeB	ot_BB/src/globals.c File Reference
5.23.	Detailed Description
5.24 EyeB	ot_BB/src/hdt.c File Reference
5.24.	Detailed Description
5.24.2	Function Documentation
	5.24.2.1 getline
	5.24.2.2 HDTClearADC
	5.24.2.3 HDTClearCAM
	5.24.2.4 HDTClearCOM
	5.24.2.5 HDTClearDRIVE
	5.24.2.6 HDTClearENCODER
	5.24.2.7 HDTClearIRTV
	5.24.2.8 HDTClearMOTOR
	5.24.2.9 HDTClearPSD
	5.24.2.10 HDTClearSERVO
	OLE NETTO TIE TO CONTOCT TO THE TENT THE TENT TO THE T
	5.24.2.11 HDTClearTable

CONTENTS ix

5.24.2.13 HDTFindTable	117
5.24.2.14 HDTLinkDRV2ENC	117
5.24.2.15 HDTLinkENC2MOT	117
5.24.2.16 HDTListEntry	118
5.24.2.17 HDTLoadADC	118
5.24.2.18 HDTLoadCAM	119
5.24.2.19 HDTLoadCOM	119
5.24.2.20 HDTLoadDRIVE	119
5.24.2.21 HDTLoadENCODER	120
5.24.2.22 HDTLoadIRTV	120
5.24.2.23 HDTLoadMOTOR	120
5.24.2.24 HDTLoadPSD	121
5.24.2.25 HDTLoadSERVO	121
5.24.2.26 HDTLoadTable	122
5.24.2.27 HDTValidate	122
5.25 EyeBot_BB/src/irtv.c File Reference	123
5.25.1 Detailed Description	123
5.25.2 Function Documentation	124
5.25.2.1 IRTVInit	124
5.25.2.2 IRTVRead	124
5.26 EyeBot_BB/src/key.c File Reference	124
5.26.1 Detailed Description	126
5.26.2 Function Documentation	126
5.26.2.1 inside	126
5.26.2.2 KEYDecode	126
5.26.2.3 KEYGet	127
5.26.2.4 KEYGetRAW	127
5.26.2.5 KEYGetRegion	127
5.26.2.6 KEYGetTM	128
5.26.2.7 KEYGetXY	128
5.26.2.8 KEYldle	128
5.26.2.9 KEYInit	129
5.26.2.10 KEYNoTouch	129
5.26.2.11 KEYRead	129

x CONTENTS

		5.26.2.12 KEYRelease	 	 			129
		5.26.2.13 KEYSetRegion	 	 		 	129
		5.26.2.14 KEYSetTM	 	 		 	130
		5.26.2.15 KEYWait	 	 			130
5.27	EyeBot	BB/src/latches.c File Reference	 	 		 	131
	5.27.1	Detailed Description	 	 			132
	5.27.2	Function Documentation	 	 			132
		5.27.2.1 OSLatchBankSetup .	 	 			132
		5.27.2.2 OSLatchCleanup	 	 			132
		5.27.2.3 OSLatchInit	 	 		 	132
		5.27.2.4 OSLatchRead	 	 		 	133
		5.27.2.5 OSLatchSetup	 	 		 	133
		5.27.2.6 OSLatchWrite	 	 		 	134
5.28	EyeBot	BB/src/lcd.c File Reference	 	 			134
	5.28.1	Detailed Description	 	 			137
	5.28.2	Function Documentation	 	 			138
		5.28.2.1 getColor	 	 		 	138
		5.28.2.2 InvertColor	 	 			138
		5.28.2.3 LCDArea	 	 		 	138
		5.28.2.4 LCDAreaInvert	 	 		 	139
		5.28.2.5 LCDClear	 	 		 	139
		5.28.2.6 LCDDrawFrame	 	 		 	139
		5.28.2.7 LCDDrawList	 	 		 	139
		5.28.2.8 LCDDrawMenu	 	 		 	140
		5.28.2.9 LCDFrame	 	 			140
		5.28.2.10 LCDGetFBInfo	 	 			140
		5.28.2.11 LCDGetList	 	 	٠.		141
		5.28.2.12 LCDGetMode	 	 	٠.		141
		5.28.2.13 LCDGetPixel	 	 	٠.		141
		5.28.2.14 LCDGetPos	 	 		 	141
		5.28.2.15 LCDInit					
		5.28.2.16 LCDInvertPixel					
		5.28.2.17 LCDLine	 	 		 	142
		5.28.2.18 LCDLineInvert	 	 			142

CONTENTS xi

	5.28.2.19 LCDList
	5.28.2.20 LCDListActiveItem
	5.28.2.21 LCDListBox
	5.28.2.22 LCDListCount
	5.28.2.23 LCDListIndex
	5.28.2.24 LCDListScrollDown
	5.28.2.25 LCDListScrollUp
	5.28.2.26 LCDMenu
	5.28.2.27 LCDMenul
	5.28.2.28 LCDMenuItem
	5.28.2.29 LCDNeedRefresh
	5.28.2.30 LCDPrintf
	5.28.2.31 LCDPutChar
	5.28.2.32 LCDPutFloat
	5.28.2.33 LCDPutFloatS
	5.28.2.34 LCDPutHex
	5.28.2.35 LCDPutHex1
	5.28.2.36 LCDPutImageRGB
	5.28.2.37 LCDPutInt
	5.28.2.38 LCDPutIntS
	5.28.2.39 LCDPutString
	5.28.2.40 LCDRefresh
	5.28.2.41 LCDRelease
	5.28.2.42 LCDResetMode
	5.28.2.43 LCDSetChar
	5.28.2.44 LCDSetList
	5.28.2.45 LCDSetMode
	5.28.2.46 LCDSetPixel
	5.28.2.47 LCDSetPos
	5.28.2.48 LCDSetPrintf
	5.28.2.49 LCDSetString
	5.28.2.50 LCDTextBar
	5.28.2.51 LCDTextColor
5.29 EyeE	Bot_BB/src/misc.c File Reference

xii CONTENTS

	5.29.1	Detailed Description
5.30	EyeBot	_BB/src/multitasking.c File Reference
	5.30.1	Detailed Description
5.31	EyeBot	_BB/src/psd_sensors.c File Reference
	5.31.1	Detailed Description
	5.31.2	Function Documentation
		5.31.2.1 PSDGet
		5.31.2.2 PSDGetRaw
		5.31.2.3 PSDInit
		5.31.2.4 PSDRelease
5.32	EyeBot	_BB/src/servos_and_motors.c File Reference
	5.32.1	Detailed Description
	5.32.2	Function Documentation
		5.32.2.1 MOTORDrive
		5.32.2.2 MOTORInit
		5.32.2.3 MOTORRelease
		5.32.2.4 QUADGetMotor
		5.32.2.5 QUADInit
		5.32.2.6 QUADRead
		5.32.2.7 QUADRelease
		5.32.2.8 QUADReset
		5.32.2.9 SERVOInit
		5.32.2.10 SERVORelease
		5.32.2.11 SERVOSet
5.33	EyeBot	_BB/src/spi.c File Reference
	5.33.1	Detailed Description
	5.33.2	Function Documentation
		5.33.2.1 checkError
		5.33.2.2 SPIInit
		5.33.2.3 SPIRead
		5.33.2.4 SPIReadDefault
		5.33.2.5 SPIRelease
		5.33.2.6 SPISend
		5.33.2.7 SPISendDefault

CONTENTS xiii

5.34 I	EyeBot_	_BB/src/sy	stem.c File	Refere	nce	)							165
į	5.34.1	Detailed [	Description										166
į	5.34.2	Function I	Documentati	on .									166
		5.34.2.1	OSError .										166
		5.34.2.2	OSInfoCPU										166
		5.34.2.3	OSInfoMem	١									167
		5.34.2.4	OSInfoMisc										167
		5.34.2.5	OSInfoProc										167
		5.34.2.6	OSMachine	ID .									167
		5.34.2.7	OSMachine	Name									168
		5.34.2.8	OSMachine	Speed	١.								168
		5.34.2.9	OSMachine	Туре									168
		5.34.2.10	OSVersion										168
5.35 I	EyeBot_	_BB/src/tin	ner.c File Re	eferenc	е								168
į	5.35.1	Detailed [	Description										169
į	5.35.2	Function I	Documentati	on .									169
		5.35.2.1	OSWait										169
5.36 I	EyeBot	_BB/src/vc	mega.c File	Refer	enc	е							170
į	5.36.1	Detailed [	Description										171
į	5.36.2	Function I	Documentati	on .									171
		5.36.2.1	VWDriveStr	aight									171
		5.36.2.2	VWDriveTu	rn									171
		5.36.2.3	VWDriveWa	ait									172
		5.36.2.4	VWInit										172

# **EyeBot\_BB Documentation**

The following describes the RoBIOS operating system library routines.

In application files use:

```
#include "eyebot.h"
```

The following libraries are available for programming the BeagleBoard in C.

In application program, include "eyebot.h" and the library will be automatically linked when calling "gcc" (refer to the example makefile provided).

Note that there are also a number of libraries available which are not listed here but are included in the EyeBot distribution (e.g. elaborate image processing library).

They can also be linked with an application program. Some of the library functions (in gray text) are not finalized yet and they are not included in this distribution.

# **Class Index**

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_HDTDevALL_t
_HDTDevCAM_t 9
_HDTDevCOM_t
_HDTDevDRIVE_t
_HDTDevENCODER_t11
_HDTDevice_t
_HDTDevIRTV_t
_HDTDevMOTOR_t
_HDTDevPSD_t
_HDTDevSERVO_t
_HDTEntry_t
_HDTTable_t
_HDTTypes_t
_process_t
_proclist_t
coord_pair_t
Structure representing the coordinates of a point
cursor_t
fbinfo_t
Hints
info_cpu_t
info_mem_t
info_misc_t
info_proc_t
Structure defining an LCD
listmenu t
m6key_box_t
money_oox_t

1	Class Index
4	Glass Index

menuitem_t	 . 24
rect_t	
Structure representing a rectangle	 . 25
screen_t	 . 25
touch_event_t	 . 26
touch_map_t	 . 27

# File Index

# 3.1 File List

Here is a list of all documented files with brief descriptions:	
EyeBot_BB/m6main-exec.h	?
EyeBot_BB/m6main-hdt.h	?
EyeBot_BB/m6main.h	?
EyeBot_BB/include/adc.h	
Header file for the ADC functions	9
EyeBot_BB/include/camera.h	
Header file for the camera functions	2
EyeBot_BB/include/eyebot.h	
· · · · · · · · · · · · · · · · · · ·	3
EyeBot_BB/include/globals.h	
	34
EyeBot_BB/include/hdt.h	
	35
EyeBot_BB/include/irtv.h	
	8
EyeBot_BB/include/key.h	
	0
EyeBot_BB/include/latches.h	_
	7
EyeBot_BB/include/lcd.h  Header file for the LCD functions	
Header file for the LCD functions	) <b>I</b>
Header file for misc functions	20
EyeBot BB/include/multitasking.h	U
Header file for multitasking functions	11
EyeBot_BB/include/params.h	
Defines main parameters	2
EyeBot BB/include/psd sensors.h	_
• – • • –	33

EyeBot_BB/include/servos_and_motors.h
Header file for the servos and motors functions
EyeBot_BB/include/spi.h
Header file for the SPI functions
EyeBot_BB/include/spi_commands.h
Defines the OP-codes for the SPI messages
EyeBot_BB/include/system.h
Header file for system functions
EyeBot_BB/include/timer.h
Header file for the timer functions
EyeBot_BB/include/types.h
Defines types
EyeBot_BB/include/vomega.h
Header file for the VW functions
EyeBot_BB/src/adc.c
Defines the ADC functions
EyeBot_BB/src/camera.c
Defines functions for the camera
EyeBot_BB/src/globals.c
Defines global variables
EyeBot_BB/src/hdt.c
Defines functions used by the HDT system
EyeBot_BB/src/irtv.c
Defines IRTV functions
EyeBot_BB/src/key.c
Defines functions for the key input
EyeBot_BB/src/latches.c
Defines functions to control latches
EyeBot_BB/src/lcd.c
Defines functions to interact with the LCD screen
EyeBot_BB/src/misc.c
Defines misc functions
EyeBot_BB/src/multitasking.c
Defines multitasking functions
EyeBot_BB/src/psd_sensors.c  Defines functions to use the PSD sensors
EyeBot_BB/src/servos_and_motors.c
• – –
Defines functions to control servos and motors
Defines fonctions for sending and receiving SPI messages 161
EyeBot_BB/src/system.c
Defines functions for the system
EyeBot BB/src/timer.c
Defines functions related to the timer
EyeBot BB/src/vomega.c
Defines the VM functions 170

# **Class Documentation**

# 4.1 \_HDTDevADC\_t Struct Reference

 $Collaboration\ diagram\ for\ \_HDTDevADC\_t:$ 



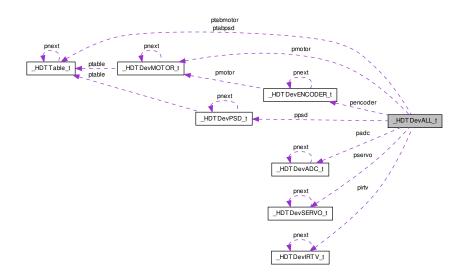
## **Public Attributes**

- struct \_HDTDevADC\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- char procname [HDT\_MAX\_NAMECHAR]
- int denom

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

## 4.2 \_HDTDevALL\_t Struct Reference

Collaboration diagram for \_HDTDevALL\_t:



#### **Public Attributes**

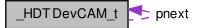
- HDTTable\_t \* ptabmotor
- HDTTable\_t \* ptabpsd
- HDTDevMOTOR\_t \* pmotor
- HDTDevENCODER\_t \* pencoder
- HDTDevPSD\_t \* ppsd
- HDTDevIRTV\_t \* pirtv
- HDTDevSERVO\_t \* pservo
- HDTDevADC\_t \* padc
- int countMOTOR
- int countENCODER
- int countPSD
- int countIRTV
- int countSERVO
- · int countADC

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

• EyeBot\_BB/m6main-hdt.c

## 4.3 \_HDTDevCAM\_t Struct Reference

Collaboration diagram for \_HDTDevCAM\_t:



#### **Public Attributes**

- struct \_HDTDevCAM\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- int regaddr
- int ucb1400io
- int width
- int height

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

• EyeBot\_BB/include/types.h

# 4.4 \_HDTDevCOM\_t Struct Reference

Collaboration diagram for \_HDTDevCOM\_t:



### **Public Attributes**

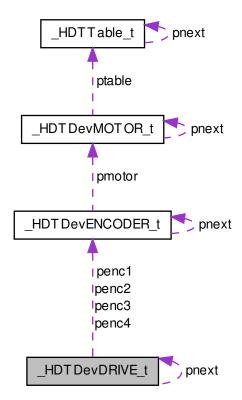
• struct \_HDTDevCOM\_t \* pnext

- char name [HDT\_MAX\_NAMECHAR]
- char devname [HDT\_MAX\_NAMECHAR]

• EyeBot\_BB/include/types.h

## 4.5 \_HDTDevDRIVE\_t Struct Reference

Collaboration diagram for \_HDTDevDRIVE\_t:



## **Public Attributes**

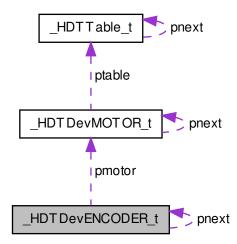
- struct \_HDTDevDRIVE\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]

- char encname1 [HDT\_MAX\_NAMECHAR]
- char encname2 [HDT\_MAX\_NAMECHAR]
- char encname3 [HDT\_MAX\_NAMECHAR]
- char encname4 [HDT\_MAX\_NAMECHAR]
- HDTDevENCODER t \* penc1
- HDTDevENCODER\_t \* penc2
- HDTDevENCODER\_t \* penc3
- HDTDevENCODER\_t \* penc4
- int drivetype
- int wheeldist1
- int axesdist
- int wheeldist2

• EyeBot\_BB/include/types.h

#### 4.6 \_HDTDevENCODER\_t Struct Reference

Collaboration diagram for \_HDTDevENCODER\_t:



### **Public Attributes**

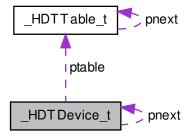
struct HDTDevENCODER t \* pnext

- char name [HDT\_MAX\_NAMECHAR]
- char motorname [HDT\_MAX\_NAMECHAR]
- HDTDevMOTOR\_t \* pmotor
- int regaddr
- · int clickspm
- · int maxspeed

• EyeBot\_BB/include/types.h

#### 4.7 \_HDTDevice\_t Struct Reference

Collaboration diagram for \_HDTDevice\_t:



#### **Public Attributes**

- struct \_HDTDevice\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- char tabname [HDT\_MAX\_NAMECHAR]
- HDTTable\_t \* ptable

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

## 4.8 \_HDTDevIRTV\_t Struct Reference

Collaboration diagram for \_HDTDevIRTV\_t:



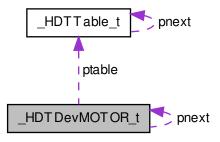
#### **Public Attributes**

- struct  $\_HDTDevIRTV\_t * pnext$
- char name [HDT\_MAX\_NAMECHAR]
- int type
- int length
- int togmask
- int invmask
- int mode
- int buffsize
- int delay

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

## 4.9 \_HDTDevMOTOR\_t Struct Reference

Collaboration diagram for \_HDTDevMOTOR\_t:



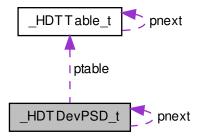
#### **Public Attributes**

- struct \_HDTDevMOTOR\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- char tabname [HDT\_MAX\_NAMECHAR]
- HDTTable\_t \* ptable
- int regaddr
- · int freq

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

## 4.10 \_HDTDevPSD\_t Struct Reference

Collaboration diagram for \_HDTDevPSD\_t:



#### **Public Attributes**

- struct \_HDTDevPSD\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- char tabname [HDT\_MAX\_NAMECHAR]
- HDTTable\_t \* ptable
- int regaddr

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

• EyeBot\_BB/include/types.h

## 4.11 \_HDTDevSERVO\_t Struct Reference

Collaboration diagram for \_HDTDevSERVO\_t:



#### **Public Attributes**

- struct \_HDTDevSERVO\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- int regaddr
- · int freq
- int mintime
- int maxtime

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

• EyeBot\_BB/include/types.h

# 4.12 \_HDTEntry\_t Struct Reference

#### **Public Attributes**

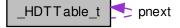
- int length
- char \* buffer

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

• EyeBot\_BB/include/types.h

#### 4.13 \_HDTTable\_t Struct Reference

Collaboration diagram for \_HDTTable\_t:



#### **Public Attributes**

- struct \_HDTTable\_t \* pnext
- char name [HDT\_MAX\_NAMECHAR]
- int size

• int \* data

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

• EyeBot\_BB/include/types.h

# 4.14 \_HDTTypes\_t Struct Reference

**Public Attributes** 

• char \* pTitle

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

• EyeBot\_BB/src/hdt.c

## 4.15 \_process\_t Struct Reference

Collaboration diagram for \_process\_t:



#### **Public Attributes**

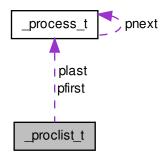
- pid\_t pid
- char name [MAX\_FILECHAR]
- struct  $\_process\_t * pnext$

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

• EyeBot\_BB/m6main-exec.c

# 4.16 \_proclist\_t Struct Reference

Collaboration diagram for proclist t:



#### **Public Attributes**

- int count
- process\_t \* pfirst
- $\bullet \ \, \text{process\_t} * \textbf{plast}$

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

• EyeBot\_BB/m6main-exec.c

# 4.17 coord\_pair\_t Struct Reference

Structure representing the coordinates of a point.

```
#include <types.h>
```

#### **Public Attributes**

- int **x**
- int **y**

### 4.17.1 Detailed Description

Structure representing the coordinates of a point.

• EyeBot\_BB/include/types.h

## 4.18 cursor\_t Struct Reference

#### **Public Attributes**

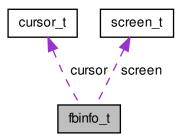
- int x
- int y
- int xmax
- int ymax

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

• EyeBot\_BB/include/types.h

## 4.19 fbinfo\_t Struct Reference

Collaboration diagram for fbinfo\_t:



## **Public Attributes**

- screen\_t screen
- cursor\_t cursor

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

#### 4.20 Hints Struct Reference

#### **Public Attributes**

- · unsigned long flags
- unsigned long functions
- · unsigned long decorations
- long inputMode
- · unsigned long status

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

• EyeBot\_BB/include/types.h

# 4.21 info\_cpu\_t Struct Reference

#### **Public Attributes**

- char name [40]
- char mhz [20]
- char arch [20]
- char bogomips [20]

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

• EyeBot BB/include/types.h

## 4.22 info\_mem\_t Struct Reference

#### **Public Attributes**

- char procnum [20]
- char total [40]
- char free [40]

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

## 4.23 info\_misc\_t Struct Reference

Public I	Attributes
----------	------------

- char uptime [20]
- char **vbatt** [20]
- int vbatt\_8

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

• EyeBot\_BB/include/types.h

# 4.24 info\_proc\_t Struct Reference

**Public Attributes** 

• char **num** [20]

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

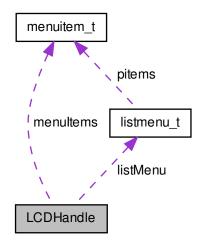
• EyeBot\_BB/include/types.h

## 4.25 LCDHandle Struct Reference

Structure defining an LCD.

#include <types.h>

Collaboration diagram for LCDHandle:



## **Public Attributes**

- int IcdNum
- Display \* d
- int **s**
- Window w
- Colormap colormap
- GC gc
- XFontStruct \* fontstruct
- int fontHeight
- int fontWidth
- int height
- int width
- int startCurPosX
- int startCurPosY
- rgb\_t fgTextColor
- rgb\_t bgTextColor
- char colorflag
- hword\_t mode
- menuitem\_t menultems [4]
- $\bullet \ \ listmenu\_t * listMenu$
- int fd
- bool X11Error

## 4.25.1 Detailed Description

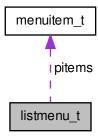
Structure defining an LCD.

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

• EyeBot\_BB/include/types.h

## 4.26 listmenu\_t Struct Reference

Collaboration diagram for listmenu\_t:



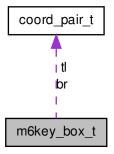
## **Public Attributes**

- char title [LCD\_LIST\_STRLENGTH]
- rgb\_t fgcol
- rgb\_t **bgcol**
- int size
- int start
- · int width
- int left
- · int scroll
- int index
- int count
- menuitem\_t \* pitems
- int no\_empty

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

# 4.27 m6key\_box\_t Struct Reference

Collaboration diagram for m6key\_box\_t:



## **Public Attributes**

- int active
- coord\_pair\_t tl
- coord\_pair\_t br

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

• EyeBot\_BB/include/types.h

## 4.28 menuitem\_t Struct Reference

## **Public Attributes**

- char label [LCD\_MENU\_STRLENGTH]
- rgb\_t fgcol
- rgb\_t bgcol
- void \* plink

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

## 4.29 rect\_t Struct Reference

Structure representing a rectangle.

```
#include <types.h>
```

#### **Public Attributes**

- int x
- int y
- · int height
- int width

## 4.29.1 Detailed Description

Structure representing a rectangle.

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

• EyeBot\_BB/include/types.h

## 4.30 screen\_t Struct Reference

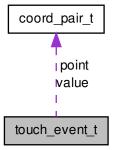
## **Public Attributes**

- int xres
- int yres
- int bpp

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

## 4.31 touch\_event\_t Struct Reference

Collaboration diagram for touch\_event\_t:



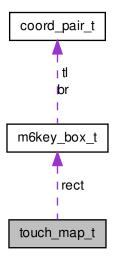
## **Public Attributes**

- coord\_pair\_t point
- coord\_pair\_t value
- int sync
- int status

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

# 4.32 touch\_map\_t Struct Reference

Collaboration diagram for touch\_map\_t:



## **Public Attributes**

- keymode\_t mode
- m6key\_box\_t rect [KEYTM\_MAX\_REGIONS]

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

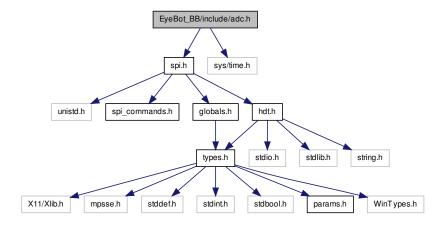
# **Chapter 5**

# **File Documentation**

# 5.1 EyeBot\_BB/include/adc.h File Reference

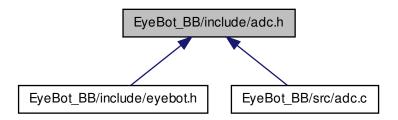
Header file for the ADC functions.

#include "spi.h" Include dependency graph for adc.h:



30 File Documentation

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



## **Functions**

- ADCHandle OSInitADC (DeviceSemantics semantics)
   Captures one single 10bit value from the specified adc channel.
- int OSADCRelease (ADCHandle handle)

Release the adc channel.

• int OSGetADC (ADCHandle adchandle)

Captures one single 10bit value from specified AD-channel. The return value is stored in the least significant bits of the 32 bit return value.

• int ConvADCSampleToVoltage (ADCHandle adchandle, char \*volt, int sample)

Convert the adc sample to voltage.

## 5.1.1 Detailed Description

Header file for the ADC functions.

**Author** 

Remi KEAT

## 5.1.2 Function Documentation

 $5.1.2.1 \quad \text{int } \textbf{ConvADCSampleToVoltage} \ ( \ \textbf{ADCHandle} \ \textit{adchandle}, \ \textbf{char} * \textit{volt}, \ \textbf{int} \ \textit{sample} \ )$ 

Convert the adc sample to voltage.

## **Parameters**

ADCHandle	adchandle : desired AD-channel
char*	volt : pointer to string
int	sample : ADC sample

Result is stored in char \*volt. Valid values: ADC0, ADC1, ADC2, ADC3

#### **Returns**

int retVal : 0: ok -1: invalid channel

## 5.1.2.2 int OSADCRelease ( ADCHandle handle )

Release the adc channel.

#### **Parameters**

ADCHandle	handle

#### Returns

int retVal: always 0

## 5.1.2.3 int OSGetADC ( ADCHandle adchandle )

Captures one single 10bit value from specified AD-channel. The return value is stored in the least significant bits of the 32 bit return value.

## Parameters

ADCHandle	handle: Handler for the adc channel
-----------	-------------------------------------

#### Returns

int retVal >0: 10 bit sampled value

-1: invalid channel

## 5.1.2.4 ADCHandle OSInitADC ( DeviceSemantics semantics )

Captures one single 10bit value from the specified adc channel.

#### **Parameters**

		_
Device-	semantics : desired ADC channel	
Semantics		

#### **Returns**

ADCHandle handle >0: Handler for the adc channel

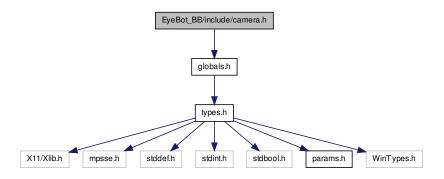
0: Initialization error

Valid values for semantics: ADC0, ADC1, ADC2, ADC3

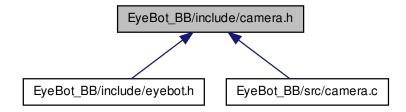
## 5.2 EyeBot\_BB/include/camera.h File Reference

Header file for the camera functions.

#include "globals.h" Include dependency graph for camera.h:



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



#### **Functions**

- CAMHandle CAMInit (DeviceSemantics semantics)

  Configure & Initialize camera.
- int CAMGetFrameRGB (CAMHandle handle, BYTE \*buf)
   Reads one full color image in RBG format, 3 bytes per pixel.

## 5.2.1 Detailed Description

Header file for the camera functions.

Author

Remi KEAT

## 5.2.2 Function Documentation

## 5.2.2.1 int CAMGetFrameRGB ( CAMHandle handle, BYTE \* buf )

Reads one full color image in RBG format, 3 bytes per pixel.

#### **Parameters**

	CAMHandle	handle : handle of the desired camera
I	BYTE*	buf : pointer to image buffer of full size (use CAMGet)

#### Returns

```
int retVal : return code
0 = success
-1 = error (camera not initialized)
```

#### 5.2.2.2 CAMHandle CAMInit ( DeviceSemantics semantics )

Configure & Initialize camera.

#### **Parameters**

Device	semantics : handle of the desired camera
Semantic	

#### **Returns**

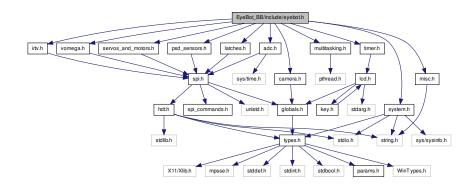
CAMHandle handle

# 5.3 EyeBot\_BB/include/eyebot.h File Reference

Header file for the EyeBot functions.

```
#include "servos_and_motors.h" #include "psd_sensors.h"
#include "timer.h" #include "latches.h" #include "multitasking.-
h" #include "system.h" #include "misc.h" #include "adc.h"
#include "irtv.h" #include "vomega.h" #include "camera.h"
```

Include dependency graph for eyebot.h:



## 5.3.1 Detailed Description

Header file for the EyeBot functions.

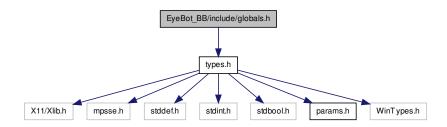
Author

Remi KEAT

# 5.4 EyeBot\_BB/include/globals.h File Reference

Header file for global variables.

#include "types.h" Include dependency graph for globals.h:



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



## **Variables**

- struct mpsse\_context \* gDeviceHandle
- LCDHandle \* gLCDHandle
- bool gLCDEnabled
- int gCurPosX
- int gCurPosY
- int gMousePosX
- int gMousePosY
- int gMouseButton
- touch\_map\_t \* gTouchMap

## 5.4.1 Detailed Description

Header file for global variables.

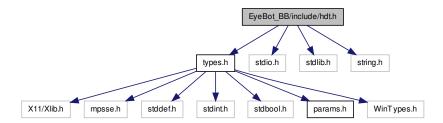
Author

Remi KEAT

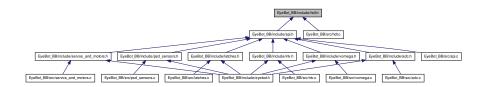
# 5.5 EyeBot\_BB/include/hdt.h File Reference

Header file for the HDT functions.

#include "types.h" #include <stdio.h> #include <stdlib.h> #include <string.h> Include dependency graph for hdt.h:



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



#### **Defines**

- #define HDT\_IDX\_TABLE 0
- #define HDT\_IDX\_PSD 1
- #define HDT\_IDX\_SERVO 2
- #define HDT IDX MOTOR 3
- #define HDT\_IDX\_ENCODER 4
- #define HDT\_IDX\_DRIVE 5
- #define HDT\_IDX\_COMPASS 6
- #define HDT\_IDX\_IRTV\_7
- #define HDT\_IDX\_CAM 8
- #define HDT\_IDX\_ADC 9
- #define **HDT\_IDX\_COM** 10
- #define HDT\_MAX\_COUNT 11
- #define HDT\_TABLE "TABLE"
- #define HDT\_PSD "PSD"
- #define HDT\_SERVO "SERVO"
- #define HDT\_MOTOR "MOTOR"
- #define HDT\_ENCODER "ENCODER"
- #define HDT\_DRIVE "DRIVE"
- #define HDT\_COMPASS "COMPASS"
- #define HDT\_IRTV "IRTV"
- #define HDT CAM "CAM"
- #define HDT ADC "ADC"
- #define HDT\_COM "SERIAL"
- #define **DIFFERENTIAL DRIVE** 0
- #define ACKERMAN\_DRIVE 1
- #define ACKERMANN\_DRIVE 1
- #define SYNCHRO DRIVE 2
- #define TRICYCLE DRIVE 3
- #define OMNI DRIVE 4
- #define HDT\_DIFF\_STR "DIFFERENTIAL"
- #define HDT\_ACKM\_STR "ACKERMANN"
- #define HDT\_OMNI\_STR "OMNI"

#### **Functions**

• int HDTValidate (char \*filename)

checks all HDT entries in given filename. will not check for specific entry (only check entry headers).

• int HDTListEntry (char \*filename, HDTEntry\_t \*deventry, int count)

Copy all entries to deventry. user need to free the allocated memory by using free(deventry->buffer). return value may be less than count.

• int HDTFindEntry (void \*hdtfile, char \*devname, HDTEntry\_t \*deventry)

finds an entry in the hdt file that matches given name and copies the entry to given structure. the newline character is replaced by null. user need to free the allocated memory by using free(deventry->buffer).

int HDTFindTable (void \*hdtfile, char \*tabname, HDTTable\_t \*tabentry)

finds a table in the hdt file that matches given name and copies the table data to given structure.

• HDTTable t \* HDTLoadTable (char \*filename, HDTDevice t \*pdevices)

load all tables needed by pdevices - if found. the return value is a pointer to the first table. the tables are in a linked list allocated with dynamic memory. use HDTClear-Table to free up the resources.

• int HDTClearTable (HDTTable t \*ptables)

Free the allocated resources for the tables created by HDTLoadTable.

• HDTDevCAM t \* HDTLoadCAM (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearCAM (HDTDevCAM t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevMOTOR\_t \* HDTLoadMOTOR (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearMOTOR (HDTDevMOTOR t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevENCODER t \* HDTLoadENCODER (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearENCODER (HDTDevENCODER t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

 int HDTLinkENC2MOT (HDTDevENCODER\_t \*pencoders, HDTDevMOTOR\_t \*pmotors)

Link the encoders to the motors.

HDTDevPSD t \* HDTLoadPSD (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearPSD (HDTDevPSD\_t \*pdevs)

Free the allocated resources for the < device> created by HDTLoad< device>.

HDTDevSERVO t \* HDTLoadSERVO (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearSERVO (HDTDevSERVO t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevDRIVE t \* HDTLoadDRIVE (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearDRIVE (HDTDevDRIVE\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

int HDTLinkDRV2ENC (HDTDevDRIVE\_t \*pdrives, HDTDevENCODER\_t \*pencoders)

Link the drives to the encoders.

• HDTDevIRTV t \* HDTLoadIRTV (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearIRTV (HDTDevIRTV\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

HDTDevADC\_t \* HDTLoadADC (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearADC (HDTDevADC\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevCOM t \* HDTLoadCOM (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearCOM (HDTDevCOM\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

## 5.5.1 Detailed Description

Header file for the HDT functions.

**Author** 

Remi KEAT

## 5.5.2 Function Documentation

## 5.5.2.1 int HDTClearADC ( HDTDevADC\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

#### **Parameters**

I	HDTDevAD-	pdevs : <device> list to be cleared</device>
	<i>C_t</i> *	

#### Returns

int retVal: always 0

## 5.5.2.2 int HDTClearCAM ( HDTDevCAM\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

#### **Parameters**

HDTDevCA-	pdevs : <device> list to be cleared</device>
<i>M_t</i> *	

## Returns

int retVal : always 0

## 5.5.2.3 int HDTClearCOM ( HDTDevCOM\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

## **Parameters**

HDTDevCO-	pdevs : <device> list to be cleared</device>
<i>M_t</i> *	

int retVal: always 0

## 5.5.2.4 int HDTClearDRIVE ( HDTDevDRIVE\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

#### **Parameters**

HDTDevDR-	pdevs : <device> list to be cleared</device>
IVE_t*	

## Returns

int retVal: always 0

## 5.5.2.5 int HDTClearENCODER ( HDTDevENCODER\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

#### **Parameters**

HDTDevEN-	pdevs : <device> list to be cleared</device>
CODER_t*	

#### Returns

int retVal: always 0

## 5.5.2.6 int HDTClearIRTV ( HDTDevIRTV\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

## **Parameters**

HDTDevIRT-	pdevs : <device> list to be cleared</device>
<i>V_t</i> *	

## Returns

int retVal: always 0

## 5.5.2.7 int HDTClearMOTOR ( HDTDevMOTOR\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

## **Parameters**

HDTDevMO-	pdevs : <device> list to be cleared</device>
TOR_t*	

#### Returns

int retVal: always 0

## 5.5.2.8 int HDTClearPSD ( HDTDevPSD\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

## **Parameters**

HDTDevPS-	pdevs : <device> list to be cleared</device>
<i>D_t</i> *	

#### **Returns**

int retVal: always 0

## $5.5.2.9 \quad \text{int HDTClearSERVO ( HDTDevSERVO\_t} * \textit{pdevs} \ )$

Free the allocated resources for the <device> created by HDTLoad<device>.

#### **Parameters**

HDTDevSE-	pdevs : <device> list to be cleared</device>	ĺ
RVO_t*		ĺ

#### Returns

int retVal: always 0

## 5.5.2.10 int HDTClearTable ( HDTTable\_t \* ptables )

Free the allocated resources for the tables created by HDTLoadTable.

#### **Parameters**

HDTTable	ptables : tables to be cleared
t*	

int retVal: always 0

## 5.5.2.11 int HDTFindEntry ( void \* hdtfile, char \* devname, HDTEntry\_t \* deventry )

finds an entry in the hdt file that matches given name and copies the entry to given structure. the newline character is replaced by null. user need to free the allocated memory by using free(deventry->buffer).

#### **Parameters**

void*	hdtfile : hdt file fopen with "rt" flag
char*	devname : name of entry to search for
HDTEntry	deventry: storage structure for the entry
t*	

#### Returns

int retVal:

-1 on failure (no entry found) [entry length] on success

```
5.5.2.12 int HDTFindTable (void * hdtfile, char * tabname, HDTTable_t * tabentry)
```

finds a table in the hdt file that matches given name and copies the table data to given structure.

#### **Parameters**

void*	hdtfile : hdt file (fopen with "rt" flag)
char*	tabname : name of table to search for
HDTTable	tabentry: storage structure for the table
<i>t</i> *	

### Returns

int retVal:

-1 on failure (no table found) [table size] on success

# 5.5.2.13 int HDTLinkDRV2ENC ( HDTDevDRIVE\_t \* pdrives, HDTDevENCODER\_t \* pencoders )

Link the drives to the encoders.

## **Parameters**

	HDTDevDR-	pdrives : list of drive methods
	IVE_t*	
ĺ	HDTDevEN-	pencoders : list of encoders
	$CODER_t*$	

#### **Returns**

int retVal:

0 on success

Negative value on failure (number of unconnected link)

# 5.5.2.14 int HDTLinkENC2MOT ( HDTDevENCODER\_t \* pencoders, HDTDevMOTOR\_t \* pmotors )

Link the encoders to the motors.

#### **Parameters**

HDTDevEN-	pencoders : list of encoders
CODER_t*	
HDTDevMO-	pmotors : list of motors
TOR_t*	

## Returns

int retVal:

0 on success

Negative value on failure (number of unconnected link)

## 5.5.2.15 int HDTListEntry ( char \* filename, HDTEntry\_t \* deventry, int count )

Copy all entries to deventry. user need to free the allocated memory by using free(deventry->buffer). return value may be less than count.

#### **Parameters**

char*	filename : name of HDT file to be checked for listing
HDTEntry	deventry: storage structure for the entry
<i>t</i> *	
int	count : number of deventry storage supplied

## Returns

int retVal:

-1 on failure

(number of entries) on success

44 File Documentation

#### 5.5.2.16 HDTDevADC\_t\* HDTLoadADC ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevADC_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

## $\textbf{5.5.2.17} \quad \textbf{HDTDevCAM\_t}* \, \textbf{HDTLoadCAM} \, ( \, \, \textbf{char} * \textit{filename}, \, \, \textbf{char} * \textit{devname} \, \, )$

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevCAM_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

## 5.5.2.18 HDTDevCOM\_t\* HDTLoadCOM ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char	*devname : device semantics

```
HDTDevCOM_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

## 5.5.2.19 HDTDevDRIVE\_t\* HDTLoadDRIVE ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

## **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevDRIVE_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

```
5.5.2.20 HDTDevENCODER_t* HDTLoadENCODER ( char * filename, char * devname )
```

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

	char*	filename : hdt file to open
Ī	char*	devname : device semantics

#### Returns

```
HDTDevENCODER_t* handle : 0x0 on failure (no <device> found) (pointer to first <device>) if found
```

#### 5.5.2.21 HDTDevIRTV\_t\* HDTLoadIRTV ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device>

are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevIRTV_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

## 5.5.2.22 HDTDevMOTOR\_t\* HDTLoadMOTOR ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevMOTOR_t* handle : 0x0 on failure (no <device> found) (pointer to first <device>) if found
```

#### 5.5.2.23 HDTDevPSD\_t\* HDTLoadPSD ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

```
HDTDevIRTV_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

## 5.5.2.24 HDTDevSERVO\_t\* HDTLoadSERVO ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevSERVO_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

## 5.5.2.25 HDTTable\_t\* HDTLoadTable ( char \* filename, HDTDevice\_t \* pdevices )

load all tables needed by pdevices - if found. the return value is a pointer to the first table. the tables are in a linked list allocated with dynamic memory. use HDTClearTable to free up the resources.

#### **Parameters**

ĺ	char*	filename : hdt file to open
	HDTDevice-	pdevices : devices with tablename in linked list
	_ <i>t</i> *	

#### Returns

```
HDTTable_t* table :
0x0 on failure (no table found)
(pointer to first table) if found
```

#### 5.5.2.26 int HDTValidate ( char \* filename )

checks all HDT entries in given filename. will not check for specific entry (only check entry headers).

#### **Parameters**

char*   filename : name of HDT file to be checked	
---	--

## Returns

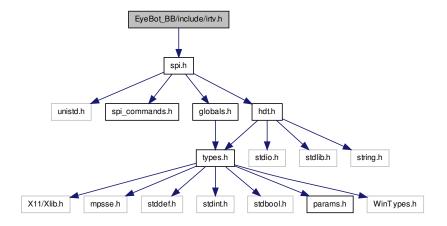
int retVal:

-1 if incorrect HDT entry found (number of entries) if otherwise

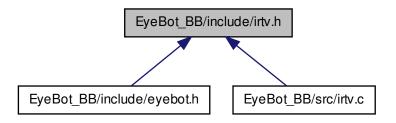
# 5.6 EyeBot\_BB/include/irtv.h File Reference

Header file for the IRTV functions.

#include "spi.h" Include dependency graph for irtv.h:



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



#### **Functions**

- int IRTVInit (DeviceSemantics semantics)
   Initializes the IR remote control decoder by calling IRTVInit() with the device name found in the corresponding HDT entry.
- int IRTVRead (void)

Reads and removes the next key code from the code buffer. Does not wait.

• void IRTVRelease (void)

Terminates the remote control decoder and releases the irtv thread.

## 5.6.1 Detailed Description

Header file for the IRTV functions.

**Author** 

Remi KEAT

### 5.6.2 Function Documentation

## 5.6.2.1 int IRTVInit ( DeviceSemantics semantics )

Initializes the IR remote control decoder by calling IRTVInit() with the device name found in the corresponding HDT entry.

#### **Parameters**

Device-	semantics
Semantics	

int retVal:

- 0 = ok
- 1 = HDT file error
- 2 = invalid or missing "IRTV" HDT entry for this semantics

## 5.6.2.2 int IRTVRead (void)

Reads and removes the next key code from the code buffer. Does not wait.

#### Returns

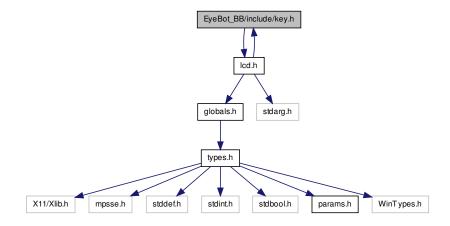
int retVal: Next code from the buffer

0 = no key

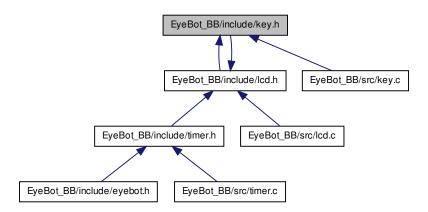
# 5.7 EyeBot\_BB/include/key.h File Reference

Header file for the key functions.

#include "lcd.h" Include dependency graph for key.h:



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



#### **Defines**

- #define KEY1 0x00000001
- #define KEY2 0x00000002
- #define KEY3 0x00000004
- #define KEY4 0x00000008
- #define KEY ESCAPE 0x80000000
- #define KEY\_LISTTL 0x40000000
- #define KEY\_LISTUP 0x20000000
- #define KEY\_LISTDN 0x10000000
- #define KEY\_LIST1 0x00000010
- #define **KEY\_LIST2** 0x00000020
- #define **KEY\_LIST3** 0x00000040
- #define **KEY\_LIST4** 0x00000080
- #define KEY\_LIST5 0x00000100
- #define **KEY\_LIST6** 0x00000200
- #define **KEY\_LIST7** 0x00000400
- #define KEY\_LIST8 0x00000800
- #define KEY\_GOIDLE 1
- #define **KEY\_NOIDLE** 0
- #define KEY\_STATE -1
- #define KEY GOIDLE 1
- #define **KEY\_NOIDLE** 0
- #define **KEY\_STATE** -1
- #define **KEYTM\_UNKNOWN** 0x00
- #define KEYTM CLASSIC 0x01

- #define **KEYTM STANDARD** 0x02
- #define KEYTM\_REGIONS 0x03
- #define KEYTM LISTMENU 0x04
- #define KEY TIMEOUT 0x00000000
- #define KEY\_INVALID 0xFFFFFFF

#### **Functions**

• int KEYInit (void)

Open the evdev device file for reading touch events. Load the key configuration file (if found), else use the hardcoded default value.

• int KEYRelease (void)

Close the evdev device file and stop checking any key touch event.

• int KEYIdle (int idle)

Enable/disable event checking procedure.

keymode\_t KEYSetTM (keymode\_t mode)

Set mode for key touch map.

keymode\_t KEYGetTM (touch\_map\_t \*\*ptouch\_map)

Get current mode and touch map (region map).

int KEYSetRegion (int index, m6key\_box\_t \*region)

Manually set region data into current touch map. Only used in KEYTM\_REGIONS mode. If region is 0x0, resets the touch map (mode becomes KEYTM\_UNKNOWN).

int KEYGetRegion (int index, m6key\_box\_t \*region)

Copy specific region data out from the current touch map. Only used in KEYTM\_RE-GIONS mode.

int KEYNoTouch (touch\_event\_t \*rawtouch)

Validate there's no touch on screen surface.

int KEYGetRAW (touch event t \*rawtouch)

Gets raw touch info - a non-blocking function. Mainly used for calibration and testing.

keycode t KEYDecode (touch event t \*rawtouch)

Decode raw touch info into a keycode based on the current touch map. Mainly used for testing.

• keycode t KEYWait (keycode t excode)

Wait for specific keys only.

keycode\_t KEYRead (void)

Read a keycode and returns. Function does not wait, thus includes KEY TIMEOUT.

keycode\_t KEYGet (void)

Wait for a touch event and return keycode (including KEY\_INVALID - undefined keycode).

coord\_pair\_t KEYGetXY (void)

Wait for a touch event and return the XY-coordinate.

• int activate\_escape (int escape)

## 5.7.1 Detailed Description

Header file for the key functions.

**Author** 

Remi KEAT

#### 5.7.2 Function Documentation

```
5.7.2.1 keycode_t KEYDecode ( touch_event_t * rawtouch )
```

Decode raw touch info into a keycode based on the current touch map. Mainly used for testing.

#### **Parameters**

```
touch_event- rawtouch : pointer to touch_event_t structure __t*
```

#### Returns

keycode t keyCode: Status of touch data (variable in rawtouch)

## 5.7.2.2 keycode\_t KEYGet ( void )

Wait for a touch event and return keycode (including KEY\_INVALID - undefined keycode).

## Returns

keycode\_t retKey : Keycode value

## 5.7.2.3 int KEYGetRAW ( touch\_event\_t \* rawtouch )

Gets raw touch info - a non-blocking function. Mainly used for calibration and testing.

## **Parameters**

```
touch_event- rawtouch : pointer to touch_event_t structure
```

## Returns

int retVal:

0 if sync signal received!

Negative value if otherwise

5.7.2.4 int KEYGetRegion (int index, m6key\_box\_t \* region)

Copy specific region data out from the current touch map. Only used in KEYTM\_REGIONS mode.

#### **Parameters**

int	index : Index for region
m6key_box-	region : Pointer to a storage for region data
_t*	

#### Returns

int retVal: 0 on success Negative value on failure

5.7.2.5 keymode\_t KEYGetTM ( touch\_map\_t \*\* ptouch\_map )

Get current mode and touch map (region map).

#### **Parameters**

touch_map-	ptouch_map : Pointer to a touch_map_t structure
_ <i>t</i> **	

#### Returns

keymode\_t retMod : Current touch map mode

5.7.2.6 coord\_pair\_t KEYGetXY (void )

Wait for a touch event and return the XY-coordinate.

## Returns

coord\_pair\_t retCoord : Coordinate pair

5.7.2.7 int KEYIdle (int idle)

Enable/disable event checking procedure.

#### **Parameters**

int	idle : user request

Valid values for idle:

- KEY\_GOIDLE deactivate event checking
- · KEY\_NOIDLE activate event checking
- KEY\_STATE request current status

int status: Idle status of event checking procedure

```
5.7.2.8 int KEYInit (void)
```

Open the evdev device file for reading touch events. Load the key configuration file (if found), else use the hardcoded default value.

#### Returns

```
int retVal: 0 on success
Negative value on failure
```

## 5.7.2.9 int KEYNoTouch ( touch\_event\_t \* rawtouch )

Validate there's no touch on screen surface.

### **Parameters**

ĺ	touch_event-	rawtouch: pointer to touch_event_t structure this is optional! only if raw
	_ <i>t</i> *	data needed! else, use 0x0!

## Returns

int retVal:

0 - being touched

1 - not touched

#### 5.7.2.10 keycode\_t KEYRead ( void )

Read a keycode and returns. Function does not wait, thus includes KEY\_TIMEOUT.

## Returns

```
keycode_t retKey : Keycode value
```

## 5.7.2.11 int KEYRelease (void)

Close the evdev device file and stop checking any key touch event.

int retVal: 0 on success Negative value on failure

## 5.7.2.12 int KEYSetRegion ( int index, m6key\_box\_t \* region )

Manually set region data into current touch map. Only used in KEYTM\_REGIONS mode. If region is 0x0, resets the touch map (mode becomes KEYTM\_UNKNOWN).

## **Parameters**

int	index : Index for region
m6key_box-	region : Pointer to a region data
_t*	

#### Returns

int retVal: 0 on success Negative value on failure

## 5.7.2.13 keymode\_t KEYSetTM ( keymode\_t mode )

Set mode for key touch map.

#### **Parameters**

I	kevmode t	mode : Requested touch map mode
	1107111000_1	mode : requested todon map mode

#### Returns

keymode\_t retMod : Current touch map mode

## 5.7.2.14 keycode\_t KEYWait ( keycode\_t excode )

Wait for specific keys only.

## **Parameters**

KAVCANA t	excode : Expected keycode values (bit XORed)
NG Y COUG L	EXCOUE : EXPECTED REVOODE VALUES (DIT XOTTED)
, _	, , ,

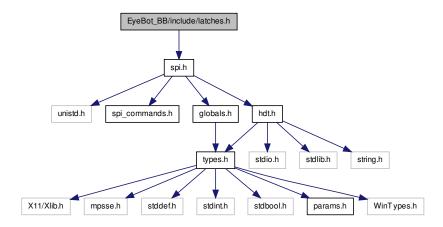
## Returns

keycode\_t retKey : Keycode value

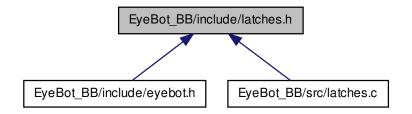
# 5.8 EyeBot\_BB/include/latches.h File Reference

Header file for the latches functions.

#include "spi.h" Include dependency graph for latches.h:



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



### **Defines**

- #define IOBANK0 0
- #define IOBANK1 1
- #define LATCH0 0
- #define **LATCH15** 15
- #define IN 0

• #define OUT 1

#### **Functions**

• int OSLatchSetup (int latchnum, int direction)

Setup the given latch as input or output.

• int OSLatchBankSetup (int banknum, int direction)

Setup the given io buffer bank as input or output.

• int OSLatchRead (int latchnum)

Read content of the selected input latch.

• int OSLatchWrite (int latchnum, int state)

Write to the selected output latch.

• int OSLatchInit (void)

Initialize the digital IO, call this before using any digital IO functions.

• int OSLatchCleanup (void)

Unmap the memory for digital IOs, call these when the digital IOs functions are no longer needed.

# 5.8.1 Detailed Description

Header file for the latches functions.

**Author** 

Remi KEAT

# 5.8.2 Function Documentation

5.8.2.1 int OSLatchBankSetup (int banknum, int direction)

Setup the given io buffer bank as input or output.

### **Parameters**

int	banknum : bank number
int	direction : signal direction

Valid values for direction:

- 0 = input
- 1 = output

Note:

• LATCH0..LATCH7 are connected to IOBANK0

• LATCH8..LATCH15 are connected to IOBANK1

#### **Returns**

```
int retVal: always 0
```

# 5.8.2.2 int OSLatchCleanup (void)

Unmap the memory for digital IOs, call these when the digital IOs functions are no longer needed.

#### Returns

```
int retVal: always 0
```

# 5.8.2.3 int OSLatchInit (void)

Initialize the digital IO, call this before using any digital IO functions.

#### **Returns**

int retVal

# Return code:

- 0 = ok
- -1 = Initialization error

# 5.8.2.4 int OSLatchRead (int latchnum)

Read content of the selected input latch.

### **Parameters**

int | latchnum : latch number to read

Return latch status:

- 0 = low
- 1 = high

#### Returns

int readValue

# 5.8.2.5 int OSLatchSetup (int latchnum, int direction)

Setup the given latch as input or output.

# Parameters

int	latchnum : latch number
int	direction : signal direction

Valid values for direction:

- 0 = input
- 1 = output

#### Returns

int retVal: always 0

# 5.8.2.6 int OSLatchWrite (int latchnum, int state)

Write to the selected output latch.

# **Parameters**

int	latchnum : latch number to write
int	state : state to be set to the selected out latch

Valid values for state:

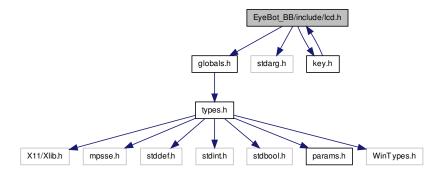
- 0 = low
- 1 = high

int retVal: always 0

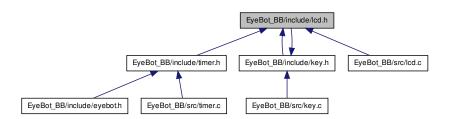
# 5.9 EyeBot\_BB/include/lcd.h File Reference

Header file for the LCD functions.

#include "globals.h" #include <stdarg.h> #include "key.h" Include dependency graph for lcd.h:



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



### **Defines**

- #define LCD\_WHITE getColor("white")
- #define LCD\_SILVER getColor("light gray")
- #define LCD\_LIGHTGRAY getColor("light gray")
- #define LCD\_LIGHTGREY getColor("light grey")
- #define LCD\_GRAY getColor("gray")

62 File Documentation

- #define LCD DARKGRAY getColor("dark gray")
- #define LCD\_DARKGREY getColor("dark grey")
- #define LCD\_BLACK getColor("black")
- #define LCD\_BLUE getColor("blue")
- #define LCD\_NAVY getColor("navy")
- #define LCD AQUA getColor("aguamarine")
- #define LCD\_CYAN getColor("cyan")
- #define LCD\_TEAL getColor("dark cyan")
- #define LCD FUCHSIA getColor("magenta")
- #define LCD\_MAGENTA getColor("magenta")
- #define LCD\_PURPLE getColor("purple")
- #define LCD RED getColor("red")
- #define LCD\_MAROON getColor("maroon")
- #define LCD\_YELLOW getColor("yellow")
- #define LCD OLIVE getColor("dark olive green")
- #define LCD\_LIME getColor("lime green")
- #define LCD\_GREEN getColor("green")
- #define LCD BGCOL TRANSPARENT 0x01
- #define LCD\_BGCOL\_NOTRANSPARENT 0x10
- #define LCD BGCOL INVERSE 0x02
- #define LCD BGCOL NOINVERSE 0x20
- #define LCD\_FGCOL\_INVERSE 0x04
- #define LCD FGCOL NOINVERSE 0x40
- #define LCD\_AUTOREFRESH 0x0001
- #define LCD\_NOAUTOREFRESH 0x0100
- #define LCD SCROLLING 0x0002
- #define LCD NOSCROLLING 0x0200
- #define LCD LINEFEED 0x0004
- #define LCD NOLINEFEED 0x0400
- #define LCD SHOWMENU 0x0008
- #define LCD\_HIDEMENU 0x0800
- #define LCD\_LISTMENU 0x0010
- #define LCD\_CLASSICMENU 0x1000
- #define LCD\_FB\_ROTATE 0x0080
- #define LCD\_FB\_NOROTATION 0x8000

#### **Functions**

• int LCDInit ()

Initialize the LCD.

int LCDClear (void)

Clear the LCD display and all display buffers.

int LCDSetMode (hword\_t mode)

Update the internal mode flag bits.

hword t LCDGetMode (void)

Get the internal mode flag bits.

int LCDResetMode (hword\_t mode)

Reset the internal mode flag bits to a previously saved mode.

• int LCDMenu (char \*string1, char \*string2, char \*string3, char \*string4)

Set menu entries in KEY\_CLASSIC mode (4-buttons). Also sets the LCD\_SHOWME-NU flag and refresh the LCD.

• int LCDMenul (int pos, char \*string, rgb\_t fgcol, rgb\_t bgcol, void \*userp)

Set specific menu entry in KEY\_CLASSIC mode (index given by pos). Color customization for specific key is now possible (fgcol/bgcol). A user-specific data can be linked to the menu using userp pointer. Will also set the LCD\_SHOWMENU flag and refresh the LCD.

• menuitem\_t \* LCDMenuItem (int index)

Return the menuitem at a given position.

• int LCDList (listmenu t \*menulist)

Setup the list menu display and update appropriate info in the listmenu\_t structure pointed by menulist (e.g. scroll, count). Will also set the LCD\_LISTMENU flag and refresh the LCD.

• int LCDSetList (listmenu t \*menulist)

Unlike LCDList(), this will blindly assign menulist to the mainlist for display. Doesn't update anything in the menulist structure, nor modify any internal flags. Useful to maintain multiple lists fo menu display.

• listmenu t \* LCDGetList (void)

Get the currently active list menu.

• menurect\_t \* LCDListBox (int pos)

Get the frame info of a specific list item in form of a menurect\_t structure.

menuitem t \* LCDListActiveItem (void)

Get the selected menuitem in the list menu – using index & start variable in listmenu\_t. Will return 0x0 (NUL) if no item is currently selected.

• rgb\_t getColor (char \*colorName)

Return the rgb\_t color from the color name.

• rgb\_t InvertColor (rgb\_t color)

Invert a RGB color.

• int LCDNeedRefresh (void)

Indicate if the LCD need to be refreshed.

int LCDArea (int x1, int y1, int x2, int y2, rgb\_t color)

Draw a color-filled rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

• int LCDSetPixel (int x, int y, rgb\_t color)

Sets the color of the pixel at (x,y) coordinate to color.

rgb\_t LCDGetPixel (int x, int y)

Get the RGB color value of the pixel at (x,y) coordinate.

• int LCDInvertPixel (int x, int y)

Bit-invert the color of the pixel at (x,y) coordinate.

int LCDLine (int x1, int y1, int x2, int y2, rgb\_t color)

Draw a color line from (x1,y1) to (x2,y2).

64 File Documentation

• int LCDLineInvert (int x1, int y1, int x2, int y2)

Draw a line from (x1,y1) to (x2,y2). The line pixels will invert the color of existing pixels.

• int LCDAreaInvert (int x1, int y1, int x2, int y2)

Draw a rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate. The pixels in the specified area will invert the color of existing pixels.

int LCDFrame (int x1, int y1, int x2, int y2, rgb\_t color)

Draw a color rectangle frame with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

• int LCDTextColor (rgb\_t fgcol, rgb\_t bgcol, char colorflags)

Set the default color for text (including background) and related flags (e.g. for transparent background).

• int LCDPrintf (const char \*format,...)

Print formatted string to LCD and refresh LCD. Cursor position is updated.

int LCDSetPrintf (int row, int column, const char \*format,...)

LCDPrintf with text position specified.

• int LCDPutChar (char c)

Write a character to LCD and refresh LCD. Cursor position is updated.

• int LCDSetChar (int row, int column, char c)

LCDPutChar with text position specified.

• int LCDPutString (char \*string)

Print string to LCD and refresh LCD. Cursor position is updated.

• int LCDSetString (int row, int column, char \*string)

LCDPutString with text position specified.

int LCDPutHex (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

• int LCDPutHex1 (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

• int LCDPutInt (int val)

Print integer to LCD and refresh LCD. Cursor position is updated.

• int LCDPutIntS (int val, int spaces)

Print integer to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

• int LCDPutFloat (float val)

Print floating-point value to LCD and refresh LCD. Cursor position is updated.

• int LCDPutFloatS (float val, int spaces, int decimals)

Print floating-point value to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

• int LCDSetPos (int row, int column)

Set the text cursor position to (row, column).

int LCDGetPos (int \*row, int \*column)

Get the text cursor position.

rect t LCDTextBar (int row, int column, int length, int fill, rgb t color)

Draw a textbar for text starting at position (row, column) until (row, column+length). The textbar will take about 25%-50% of text height & width to draw its frame. The fill parameter will define how much of the text bar should be 'filled' with color (like a progress bar).

• int LCDRelease ()

Release the LCD.

· int LCDRefresh (void)

Refresh the screen (i.e write display buffers to the framebuffer device).

int LCDGetFBInfo (fbinfo t \*pinfo)

Get display information and save to structure pointed by pinfo. Cursor info needs LC-Dlnit() for textsize.

int LCDListCount (void)

Get the number of list items supported by the current display (text) configuration. - This includes the item for title bar - thus, different from count variable in listmenu\_t as updated by an LCDList() call.

• int LCDListIndex (int index)

Set the list index.

• int LCDListScrollUp (void)

Scrolls the list display up. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

• int LCDListScrollDown (void)

Scrolls the list display down. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

int LCDPutImageRGB (int xpos, int ypos, int xsize, int ysize, byte\_t \*data)

Place a RGB color image (24bpp) at (xpos,ypos) position on the LCD screen.

# 5.9.1 Detailed Description

Header file for the LCD functions.

Author

Remi KEAT

# 5.9.2 Function Documentation

5.9.2.1 rgb\_t getColor ( char \* colorName )

Return the rgb\_t color from the color name.

char*	colorName
-------	-----------

rgb\_t color

# 5.9.2.2 rgb\_t InvertColor ( rgb\_t color )

Invert a RGB color.

#### **Parameters**

```
rgb_t | color : RGB color value
```

#### Returns

rgb\_t color : RGB color value

# 5.9.2.3 int LCDArea ( int x1, int y1, int x2, int y2, $rgb_t color$ )

Draw a color-filled rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

#### **Parameters**

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel
rgb_t	color : RGB fill color value

#### Returns

int retVal: always 0

# 5.9.2.4 int LCDAreaInvert ( int x1, int y1, int x2, int y2 )

Draw a rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate. The pixels in the specified area will invert the color of existing pixels.

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel

int retVal: always 0

5.9.2.5 int LCDClear (void)

Clear the LCD display and all display buffers.

#### **Returns**

int retVal: always 0

5.9.2.6 int LCDFrame ( int x1, int y1, int x2, int y2, rgb\_t color )

Draw a color rectangle frame with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

#### **Parameters**

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel
rgb_t	color : RGB fill color value

#### **Returns**

int retVal: always 0

5.9.2.7 int LCDGetFBInfo ( fbinfo\_t \* pinfo )

Get display information and save to structure pointed by pinfo. Cursor info needs LCD-Init() for textsize.

### **Parameters**

fbinfo\_t\* pinfo: Pointer to storage for screen & cursor info

# Returns

int retVal

0 on success

Negative value on failure

File Documentation

```
5.9.2.8 listmenu_t* LCDGetList ( void )
```

Get the currently active list menu.

#### Returns

```
listmenu_t* retListMenu : Pointer to listmenu_t structure
```

```
5.9.2.9 hword_t LCDGetMode ( void )
```

Get the internal mode flag bits.

#### Returns

hword\_t mode : Current mode flag bits

```
5.9.2.10 rgb_t LCDGetPixel ( int x, int y )
```

Get the RGB color value of the pixel at (x,y) coordinate.

#### **Parameters**

int	x : X-coordinate of the pixel
int	y: Y-coordinate of the pixel

# Returns

```
rgb_t color : RGB color value
```

```
5.9.2.11 int LCDGetPos ( int * row, int * column )
```

Get the text cursor position.

### **Parameters**

int*	row : Pointer to cursor row index
int*	column : Pointer to cursor column index

### Returns

int retVal: always 0

5.9.2.12 int LCDInit ( )

Initialize the LCD.

int retVal: always 0

# 5.9.2.13 int LCDInvertPixel ( int x, int y )

Bit-invert the color of the pixel at (x,y) coordinate.

#### **Parameters**

in	x : X-coordinate of the pixel
in	y: Y-coordinate of the pixel

## Returns

int retVal: always 0

# 5.9.2.14 int LCDLine ( int x1, int y1, int x2, int y2, $rgb_t color$ )

Draw a color line from (x1,y1) to (x2,y2).

#### **Parameters**

int	x1 : X-coordinate of first pixel
int	y1 : Y-coordinate of first pixel
int	x2 : X-coordinate of second pixel
int	y2 : Y-coordinate of second pixel
rgb_t	color : RGB color value for the pixel

# Returns

int retVal: always 0

# 5.9.2.15 int LCDLineInvert (int x1, int y1, int x2, int y2)

Draw a line from (x1,y1) to (x2,y2). The line pixels will invert the color of existing pixels.

int	x1 : X-coordinate of first pixel
int	y1 : Y-coordinate of first pixel
int	x2 : X-coordinate of second pixel
int	y2 : Y-coordinate of second pixel

70 File Documentation

#### Returns

int retVal: always 0

```
5.9.2.16 int LCDList ( listmenu_t * menulist )
```

Setup the list menu display and update appropriate info in the <a href="listmenu\_t">listmenu\_t</a> structure pointed by menulist (e.g. scroll, count). Will also set the LCD\_LISTMENU flag and refresh the LCD.

#### **Parameters**

```
listmenu_t* menulist : Listmenu to be used for display
```

#### Returns

int retVal: always 0

```
5.9.2.17 menuitem_t* LCDListActiveItem ( void )
```

Get the selected menuitem in the list menu – using index & start variable in listmenu\_t. Will return 0x0 (NUL) if no item is currently selected.

## Returns

```
menuitem_t* retMenuItem : Pointer to a menuitem_t structure
```

```
5.9.2.18 menurect_t* LCDListBox ( int pos )
```

Get the frame info of a specific list item in form of a menurect\_t structure.

#### **Parameters**

```
int pos : Index of list item
```

#### Returns

```
menurect_t* retMenuRect : Pointer to a menurect_t structure
```

### 5.9.2.19 int LCDListCount (void)

Get the number of list items supported by the current display (text) configuration. This includes the item for title bar - thus, different from count variable in listmenu\_t as updated by an LCDList() call.

int listCount: Number of list items (including title box)

5.9.2.20 int LCDListIndex (int index)

Set the list index.

#### **Parameters**

int	index . Listindex
1111	index : List index
	I make the second secon

#### Returns

int retVal: List index

## 5.9.2.21 int LCDListScrollDown (void)

Scrolls the list display down. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

#### Returns

int retVal: always 0

## 5.9.2.22 int LCDListScrollUp (void)

Scrolls the list display up. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

## **Returns**

int retVal: always 0

5.9.2.23 int LCDMenu ( char \* string1, char \* string2, char \* string3, char \* string4 )

Set menu entries in KEY\_CLASSIC mode (4-buttons). Also sets the LCD\_SHOWMENU flag and refresh the LCD.

char*	string1 : Menu entry for KEY1 in classic mode
char*	string2 : Menu entry for KEY2 in classic mode
char*	string3 : Menu entry for KEY3 in classic mode
char*	string4 : Menu entry for KEY4 in classic mode

int retVal: always 0

5.9.2.24 int LCDMenul ( int pos, char \* string, rgb\_t fgcol, rgb\_t bgcol, void \* userp )

Set specific menu entry in KEY\_CLASSIC mode (index given by pos). Color customization for specific key is now possible (fgcol/bgcol). A user-specific data can be linked to the menu using userp pointer. Will also set the LCD\_SHOWMENU flag and refresh the LCD.

#### **Parameters**

int	pos : Select menu entry in classic mode
char*	string: Menu entry for the key at specified index
rgb_t	fgcol: Textcolor for the menu
rgb_t	bgcol : Background color for the menu
void*	userp: A general purpose pointer for user-specific data

#### Returns

int retVal: always 0

5.9.2.25 menuitem\_t\* LCDMenuItem ( int index )

Return the menuitem at a given position.

# Parameters

int	index : position of the menuitem

#### Returns

menuitem\_t\* menuItem

5.9.2.26 int LCDNeedRefresh (void)

Indicate if the LCD need to be refreshed.

# Returns

int retVal: non-null value indicate that the LCD need to be refreshed

5.9.2.27 int LCDPrintf (const char \* format, ...)

Print formatted string to LCD and refresh LCD. Cursor position is updated.

## **Parameters**

	char* format : Formatted string
L CONST	Char* format : Formatied string
001101	Onder Torride : 1 Ormation String

#### Returns

int retVal: always 0

# 5.9.2.28 int LCDPutChar (char c)

Write a character to LCD and refresh LCD. Cursor position is updated.

## **Parameters**

,	
cnar	c : Character to be displayed
Ullai	C. Ondracter to be displayed

#### Returns

int retVal: always 0

# 5.9.2.29 int LCDPutFloat (float val)

Print floating-point value to LCD and refresh LCD. Cursor position is updated.

#### **Parameters**

ĺ	int	val : Floating-point value to be displayed
	1111	vai . I loating point value to be displayed

# Returns

int retVal: always 0

# 5.9.2.30 int LCDPutFloatS ( float val, int spaces, int decimals )

Print floating-point value to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

int	val : Floating-point value to be displayed
int	spaces : Text space for the integer
int	decimals: Number of decimal points to display

int retVal: always 0

# 5.9.2.31 int LCDPutHex (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

#### **Parameters**

int	Lval : Hay number to be displayed
1111	val : Hex number to be displayed
	Tail Trion Harrison to So allegia ou

#### Returns

int retVal: always 0

# 5.9.2.32 int LCDPutHex1 (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

#### **Parameters**

int   val : Hex number to be displayed
--

#### Returns

int retVal: always 0

# 5.9.2.33 int LCDPutImageRGB ( int xpos, int ypos, int xsize, int ysize, byte\_t \* data )

Place a RGB color image (24bpp) at (xpos,ypos) position on the LCD screen.

## **Parameters**

int	xpos : X-coordinate of top-left image position
int	ypos: Y-coordinate of top-left image position
int	xsize : Image width
int	ysize : Image height
byte_t*	data : Pointer to image data (24-bit per pixel)

#### Returns

int retVal: always 0

5.9.2.34 int LCDPutInt (int val)

Print integer to LCD and refresh LCD. Cursor position is updated.

#### **Parameters**

```
int | val : Integer to be displayed
```

#### **Returns**

int retVal: always 0

5.9.2.35 int LCDPutIntS (int val, int spaces)

Print integer to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

#### **Parameters**

int	val : Integer to be displayed
int	spaces : Text space for the integer

#### **Returns**

int retVal: always 0

5.9.2.36 int LCDPutString ( char \* string )

Print string to LCD and refresh LCD. Cursor position is updated.

#### **Parameters**

char*   string : String to be displayed
---

## **Returns**

int retVal: always 0

5.9.2.37 int LCDRefresh (void)

Refresh the screen (i.e write display buffers to the framebuffer device).

#### Returns

int retVal: always 0

```
5.9.2.38 int LCDRelease ( )
```

Release the LCD.

#### Returns

int retVal : always 0

## 5.9.2.39 int LCDResetMode ( hword\_t mode )

Reset the internal mode flag bits to a previously saved mode.

#### **Parameters**

hward timada : Mada tia
I IIWOIG I IIIOGE I WOGE IIAC
hword t mode: Mode flag

#### Returns

int retVal: always 0

5.9.2.40 int LCDSetChar (int row, int column, char c)

LCDPutChar with text position specified.

#### **Parameters**

int	row : Cursor position
int	column : Cursor position
char	c : Character to be displayed

#### Returns

int retVal: always 0

# 5.9.2.41 int LCDSetList ( listmenu\_t \* menulist )

Unlike LCDList(), this will blindly assign menulist to the mainlist for display. Doesn't update anything in the menulist structure, nor modify any internal flags. Useful to maintain multiple lists fo menu display.

listmenu t*   menulist : Listmenu to be used for display	listmenu t*	menulist : Listmenu to be used for display
--	-------------	--

int retVal: always 0

5.9.2.42 int LCDSetMode ( hword\_t mode )

Update the internal mode flag bits.

#### **Parameters**

_			
	I I		
- 1	nwora t	mode : LCD Mode flag	
- 1	c.a_t	mode . Lob mode nag	

## Returns

int retVal: always 0

5.9.2.43 int LCDSetPixel (int x, int y, rgb\_t color)

Sets the color of the pixel at (x,y) coordinate to color.

## **Parameters**

int	x : X-coordinate of the pixel
int	y: Y-coordinate of the pixel
rgb_t	color : RGB color value for the pixel

# Returns

int retVal: always 0

5.9.2.44 int LCDSetPos (int row, int column)

Set the text cursor position to (row, column).

int	row : Text cursor row index
int	column : Text cursor column index

int retVal: always 0

5.9.2.45 int LCDSetPrintf ( int row, int column, const char \* format, ... )

LCDPrintf with text position specified.

#### **Parameters**

int	row : Cursor position
int	column : Cursor position
const	char* format : Formatted string

#### Returns

int retVal: always 0

5.9.2.46 int LCDSetString (int row, int column, char \* string)

LCDPutString with text position specified.

### **Parameters**

int	row : Cursor position
int	column : Cursor position
char*	c : String to be displayed

### Returns

int retVal: always 0

5.9.2.47 rect\_t LCDTextBar ( int row, int column, int length, int fill, rgb\_t color )

Draw a textbar for text starting at position (row, column) until (row, column+length). - The textbar will take about 25%-50% of text height & width to draw its frame. The fill parameter will define how much of the text bar should be 'filled' with color (like a progress bar).

int	row : Start text cursor position
int	column : Start text cursor position
int	length: Text length of the bar
int	fill : Percentage of textbar to be filled
rgb_t	color : Fill color for the textbar

rect\_t rect : rect\_t structure for the textbar's frame

5.9.2.48 int LCDTextColor ( rgb\_t fgcol, rgb\_t bgcol, char colorflags )

Set the default color for text (including background) and related flags (e.g. for transparent background).

#### **Parameters**

rgb_t	fgcol : Default color for text
rgb_t	bgcol : Default color for text background
char	colorflags : Mode flag for text color

# Valid value for colorflags:

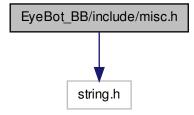
- LCD\_BGCOL\_TRANSPARENT
- LCD\_BGCOL\_INVERSE
- LCD\_FGCOL\_INVERSE
- LCD\_BGCOL\_NOTRANSPARE
- LCD\_BGCOL\_NOINVERSE
- LCD\_FGCOL\_NOINVERSE

int retVal: always 0

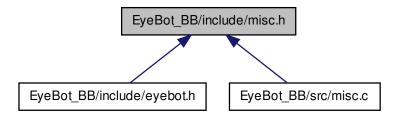
# 5.10 EyeBot\_BB/include/misc.h File Reference

Header file for misc functions.

#include <string.h> Include dependency graph for misc.h:



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



# **Functions**

• void **strcpy\_n** (char \*\_\_dest, const char \*\_\_src, size\_t \_\_n)

# 5.10.1 Detailed Description

Header file for misc functions.

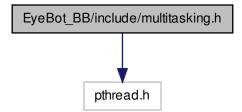
**Author** 

Remi KEAT

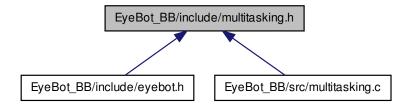
# 5.11 EyeBot\_BB/include/multitasking.h File Reference

Header file for multitasking functions.

#include < pthread.h > Include dependency graph for multitasking.h:



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



# 5.11.1 Detailed Description

Header file for multitasking functions.

**Author** 

Remi KEAT

# 5.12 EyeBot\_BB/include/params.h File Reference

Defines main parameters.

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



#### **Defines**

- #define VEHICLE 1
- #define PLATFORM 2
- #define WALKER 3
- #define **DEBUG** 1
- #define NUMBER TRY 10
- #define HDT\_MAX\_NAMECHAR 80
- #define LCD\_MENU\_STRLENGTH 32 /\* for storage declaration \*/
- #define LCD\_LIST\_STRLENGTH 64 /\* for storage declaration \*/
- #define MENU\_HEIGHT 38
- #define KEYTM\_MAX\_REGIONS 32
- #define VERSION "1.0"
- #define MACHINE SPEED 1000000000
- #define MACHINE\_TYPE VEHICLE
- #define MACHINE\_NAME "EyeBot"
- #define ID 1
- #define CPU ARCH "ARM"
- #define CPU\_BOGOMIPS "?"
- #define CPU\_MHZ "1000"
- #define CPU\_NAME "AM37x 1GHz ARM Cortex-A8 compatible"
- #define LIBM6OS\_VERSION "1.0"
- #define LCD\_TYPE BEAGLETOUCH
- #define LCD HEIGHT 272
- #define LCD\_WIDTH 480
- #define LCD\_MAX\_LIST\_ITEM 8
- #define **HDT\_FILE** "hdt.txt"
- #define HDT\_MAX\_PATHCHAR 256
- #define HDT\_MAX\_FILECHAR 40
- #define HDT\_MAX\_READBUFF 128

# 5.12.1 Detailed Description

Defines main parameters.

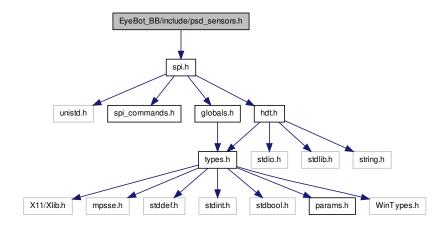
Author

Remi KEAT

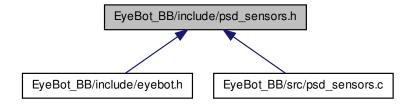
# 5.13 EyeBot\_BB/include/psd\_sensors.h File Reference

Header file for the PSD sensors functions.

#include "spi.h" Include dependency graph for psd\_sensors.h:



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



#### **Functions**

PSDHandle PSDInit (DeviceSemantics semantics)

Initialize single PSD with given semantics. Up to 8 PSDs can be initialized.

• int PSDGetRaw (PSDHandle psd)

Delivers raw-data measured by the selected PSD.

• int PSDGet (PSDHandle psd)

Delivers actual timestamp or distance measured by the selected PSD. If the raw reading is out of range for the given sensor, PSD\_OUT\_OF\_RANGE (=9999) is returned.

• int PSDRelease (PSDHandle psd)

Stops measurings and releases a PSD.

# 5.13.1 Detailed Description

Header file for the PSD sensors functions.

Author

Remi KEAT

## 5.13.2 Function Documentation

# 5.13.2.1 int PSDGet ( PSDHandle psd )

Delivers actual timestamp or distance measured by the selected PSD. If the raw reading is out of range for the given sensor, PSD\_OUT\_OF\_RANGE (=9999) is returned.

# Parameters

PSDHandle 1 4 1	psd : the number of the psd to read

#### Returns

int retVal: actual distance in mm (converted through internal table)

# 5.13.2.2 int PSDGetRaw ( PSDHandle psd )

Delivers raw-data measured by the selected PSD.

PSDHandle	psd: Handle of the psd to read
-----------	--------------------------------

int readVal: actual raw-data (not converted)

# 5.13.2.3 PSDHandle PSDInit ( DeviceSemantics semantics )

Initialize single PSD with given semantics. Up to 8 PSDs can be initialized.

#### **Parameters**

Device	semantics : unique definition for desired PSD
Semantic	

## Returns

PSDHandle psdHandle: unique handle for all further operations

# 5.13.2.4 int PSDRelease ( PSDHandle psd )

Stops measurings and releases a PSD.

#### **Parameters**

<i>PSDHandle</i>	psd

#### **Returns**

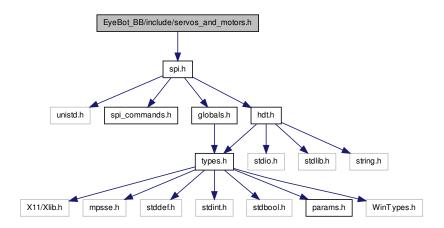
int retVal: always 0

# 5.14 EyeBot\_BB/include/servos\_and\_motors.h File Reference

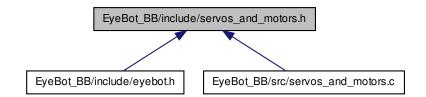
Header file for the servos and motors functions.

86 File Documentation

#include "spi.h" Include dependency graph for servos\_and\_motors.h:



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



# **Functions**

- SERVOHandle SERVOInit (DeviceSemantics semantics)
  - Initialize given servo.
- int SERVORelease (SERVOHandle handle)
  - Release given servos.
- int SERVOSet (SERVOHandle handle, int angle)
  - Set the given servos to the same given angle.
- MOTORHandle MOTORInit (DeviceSemantics semantics)
  - Initialize given motor.
- int MOTORRelease (MOTORHandle handle)

Release given motor.

• int MOTORDrive (MOTORHandle handle, int speed)

Set the given motors to the same given speed.

• long QUADRead (QUADHandle handle)

Read actual Quadrature-Decoder counter, initially zero.

• int QUADReset (QUADHandle handle)

Reset one or more Quadrature-Decoder.

• int QUADRelease (QUADHandle handle)

Release one or more Quadrature-Decoder.

## 5.14.1 Detailed Description

Header file for the servos and motors functions.

Author

Remi KEAT

#### 5.14.2 Function Documentation

#### 5.14.2.1 int MOTORDrive (MOTORHandle handle, int speed)

Set the given motors to the same given speed.

## Parameters

MOTOR- Handle	
	speed : motor speed in percent

Valid values for speed:

- -100 to 100 (full backward to full forward)
- 0 for full stop

### Returns

int retVal: always 0

#### 5.14.2.2 MOTORHandle MOTORInit ( DeviceSemantics semantics )

Initialize given motor.

Device-	semantics
Semantics Generated on Sat So	ep 14 2013 08:41:42 for EyeBot_BB by Doxygen

MOTORHandle motorHandle

# 5.14.2.3 int MOTORRelease ( MOTORHandle handle )

Release given motor.

#### **Parameters**

MOTOR-	handle
Handle	

## Returns

int retVal: always 0

# 5.14.2.4 long QUADRead ( QUADHandle handle )

Read actual Quadrature-Decoder counter, initially zero.

#### **Parameters**

QUAD-	handle : ONE decoder-handle
Handle	

#### Returns

long value of the encoder

# 5.14.2.5 int QUADRelease ( QUADHandle handle )

Release one or more Quadrature-Decoder.

### **Parameters**

QUAD-	handle : logical-or of decoder-handles to be released
Handle	

## Returns

int retVal : 0 = ok -1 = error wrong handle 5.14.2.6 int QUADReset ( QUADHandle handle )

Reset one or more Quadrature-Decoder.

#### **Parameters**

QUAD-	handle: logical-or of decoder-handles to be reseted
Handle	

#### Returns

int retVal : 0 = ok -1 = error wrong handle

5.14.2.7 SERVOHandle SERVOInit ( DeviceSemantics semantics )

Initialize given servo.

#### **Parameters**

Device-	semantics
Semantics	

#### Returns

SERVOHandle servoHandle

5.14.2.8 int SERVORelease ( SERVOHandle handle )

Release given servos.

# **Parameters**

SERVO-	handle
Handle	

# Returns

int retVal: always 0

5.14.2.9 int SERVOSet ( SERVOHandle handle, int angle )

Set the given servos to the same given angle.

#### **Parameters**

SERVO-	handle
Handle	
int	angle : valid values = 0-360

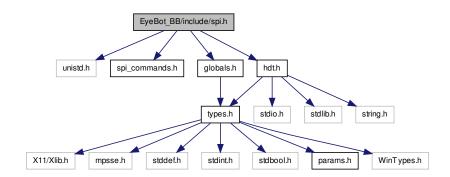
#### Returns

int retVal: always 0

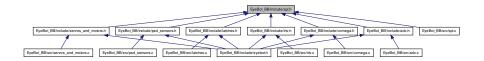
# 5.15 EyeBot\_BB/include/spi.h File Reference

Header file for the SPI functions.

#include <unistd.h> #include "spi\_commands.h" #include
"globals.h" #include "hdt.h" Include dependency graph for spi.h:



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



## **Functions**

- SPIHandle SPIInit (int deviceNumber)

  Initialize the SPI device.
- int SPIRelease (SPIHandle spiHandle)

Release the SPI device.

int SPISend (SPIHandle spiHandle, size\_t length, const uint8\_t data[])
 Send a SPI message.

• int SPISendDefault (size\_t length, const uint8\_t data[])

Send a SPI message on the default SPI device.

• int SPIRead (SPIHandle spiHandle, size\_t length, uint8\_t \*data[])

Read a SPI message.

• int SPIReadDefault (size\_t length, uint8\_t \*data[])

Read a SPI message on the default SPI device.

# 5.15.1 Detailed Description

Header file for the SPI functions.

Author

Remi KEAT

## 5.15.2 Function Documentation

5.15.2.1 SPIHandle SPIInit (int deviceNumber)

Initialize the SPI device.

#### **Parameters**

int deviceNumber
------------------

# Returns

SPIHandle spiHandle

5.15.2.2 int SPIRead ( SPIHandle <code>spiHandle</code>, <code>size\_t</code> <code>length</code>, <code>uint8\_t \* data[]</code> )

Read a SPI message.

#### **Parameters**

ĺ	SPIHandle	spiHandle
	size_t	length
	uint8_t*	data[]

#### **Returns**

int retVal: always 0

92 File Documentation

5.15.2.3 int SPIReadDefault ( size\_t length, uint8\_t \* data[] )

Read a SPI message on the default SPI device.

## **Parameters**

size_t	length
uint8_t*	data[]

#### Returns

int retVal: always 0

5.15.2.4 int SPIRelease (SPIHandle spiHandle)

Release the SPI device.

#### **Parameters**

SPIHandle spiHandle
---------------------

#### Returns

int retVal: always 0

5.15.2.5 int SPISend ( SPIHandle spiHandle, size\_t length, const uint8\_t data[] )

Send a SPI message.

### **Parameters**

SPIHandle	spiHandle
size_t	length
const	uint8_t data[]

#### Returns

int retVal: always 0

5.15.2.6 int SPISendDefault ( size\_t length, const uint8\_t data[] )

Send a SPI message on the default SPI device.

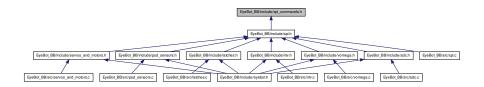
size_t	length
const	uint8_t data[]

int retVal: always 0

# 5.16 EyeBot\_BB/include/spi\_commands.h File Reference

Defines the OP-codes for the SPI messages.

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



# **Defines**

- #define SPIServoInitCmd 0x01
- #define SPIServoReleaseCmd 0x02
- #define SPIServoSetCmd 0x03
- #define SPIMotorInitCmd 0x04
- #define SPIMotorReleaseCmd 0x05
- #define SPIMotorSetCmd 0x06
- #define SPIReadEncoderCmd 0x07
- #define SPIPSDGetCmd 0x08
- #define SPILatchSetupCmd 0x09
- #define SPILatchBankSetupCmd 0x0A
- #define SPILatchReadCmd 0x0B
- #define SPILatchWriteCmd 0x0C
- #define SPIResetEncoderCmd 0x0D
- #define SPIVWInitCmd 0x0E
- #define SPIVWDriveStraightCmd 0x0F
- #define SPIVWDriveWaitCmd 0x10
- #define SPIWheelDist1Param 0x01
- #define SPIWheelDist2Param 0x02
- #define SPIAxesDistParam 0x03
- #define SPIEncoderParam 0x04
- #define SPIDistanceParam 0x05
- #define SPISpeedParam 0x06

# 5.16.1 Detailed Description

Defines the OP-codes for the SPI messages.

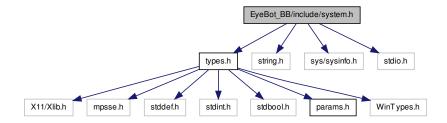
**Author** 

Remi KEAT

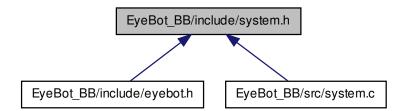
# 5.17 EyeBot\_BB/include/system.h File Reference

Header file for system functions.

#include "types.h" #include "string.h" #include <sys/sysinfo.h> #include <stdio.h> Include dependency graph for system.h:



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



# **Functions**

• char \* execute (char \*command)

char \* OSVersion (void)

Returns string containing running RoBIOS version.

int OSMachineSpeed (void)

Inform the user how fast the processor runs.

int OSMachineType (void)

Inform the user in which environment the program runs.

char \* OSMachineName (void)

Inform the user with which name the Eyebot is titled.

unsigned char OSMachineID (void)

Inform the user with which ID the Eyebot is titled.

int OSError (char \*msg, int number, bool deadend)

Print message and number to display then stop processor (deadend) or wait for key.

• int OSInfoCPU (info\_cpu\_t \*infoCPU)

Collects infos about the CPU – name, speed, architecture and bogusMips.

• int OSInfoMem (info\_mem\_t \*infoMem)

Collects infos about the memory.

• int OSInfoProc (info\_proc\_t \*infoProc)

Collects infos about processes.

• int OSInfoMisc (info\_misc\_t \*infoMisc)

Collects system's miscellaneous infos - uptime, vbatt.

# 5.17.1 Detailed Description

Header file for system functions.

Author

Remi KEAT

### 5.17.2 Function Documentation

5.17.2.1 int OSError ( char \* msg, int number, bool deadend )

Print message and number to display then stop processor (deadend) or wait for key.

### **Parameters**

ĺ	char*	msg : pointer to message
	int	number : int number
	BOOL	deadend : switch to choose deadend or keywait

Valid values are:

- 0 = no deadend
- 1 = deadend

int retVal : Always 0

# 5.17.2.2 int OSInfoCPU ( info\_cpu\_t \* infoCPU )

Collects infos about the CPU - name, speed, architecture and bogusMips.

### **Parameters**

```
info_cpu_t* InfoCPU : pointer to a structure (info_cpu_t) containing the cpu infos
```

# Returns

int retVal: always 0

# 5.17.2.3 int OSInfoMem ( info\_mem\_t \* infoMem )

Collects infos about the memory.

### **Parameters**

info_mem_t*	infoMem: pointer to a structure (info_mem_t) which contains the mem-
	ory infos

# Returns

int retVal: always 0

# 5.17.2.4 int OSInfoMisc ( info\_misc\_t \* infoMisc )

Collects system's miscellaneous infos – uptime, vbatt.

### **Parameters**

```
info_misc_t infoMisc : pointer to a structure (info_misc_t) which contains the misc infos
```

### Returns

int retVal: always 0

# 5.17.2.5 int OSInfoProc ( info\_proc\_t \* infoProc )

Collects infos about processes.

### **Parameters**

info\_proc\_t infoProc : pointer to a structure (info\_proc\_t) which contains the process infos

# Returns

int retVal: always 0

5.17.2.6 unsigned char OSMachineID (void)

Inform the user with which ID the Eyebot is titled.

**Returns** 

unsigned char ID: ID of actual Eyebot

5.17.2.7 char\* OSMachineName (void)

Inform the user with which name the Eyebot is titled.

Returns

char\* machineName : Name of actual Eyebot

5.17.2.8 int OSMachineSpeed (void)

Inform the user how fast the processor runs.

Returns

int speed: actual clockrate of CPU in Hz

5.17.2.9 int OSMachineType (void)

Inform the user in which environment the program runs.

Returns

int machineType: Type of used hardware

Valid values are: VEHICLE, PLATFORM, WALKER

5.17.2.10 char\* OSVersion (void)

Returns string containing running RoBIOS version.

### Returns

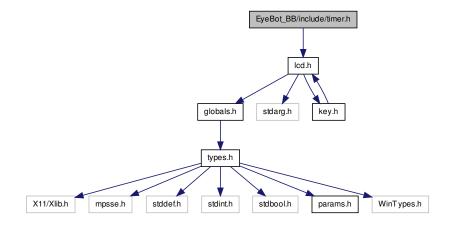
char\* version : OS version

Example: "3.1b"

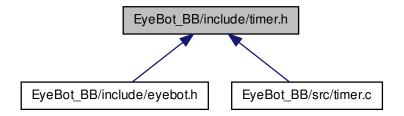
# 5.18 EyeBot\_BB/include/timer.h File Reference

Header file for the timer functions.

#include "lcd.h" Include dependency graph for timer.h:



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



# **Functions**

• int OSWait (int n)

Busy loop for n\*1/100 seconds.

# 5.18.1 Detailed Description

Header file for the timer functions.

Author

Remi KEAT

# 5.18.2 Function Documentation

5.18.2.1 int OSWait ( int n )

Busy loop for n\*1/100 seconds.

# **Parameters**

int n: time to wait

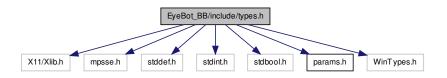
100

int retVal: always 0

# 5.19 EyeBot\_BB/include/types.h File Reference

### Defines types.

#include <X11/Xlib.h> #include <mpsse.h> #include <stddef.h> #include <stdint.h> #include <stdbool.h> #include
"params.h" #include "WinTypes.h" Include dependency graph for types.h:



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



# Classes

- struct Hints
- struct screen t
- · struct cursor\_t
- struct fbinfo\_t
- struct info\_cpu\_t
- · struct info mem t
- struct info\_proc\_t
- struct info\_misc\_t
- struct coord\_pair\_t

Structure representing the coordinates of a point.

- struct m6key\_box\_t
- · struct touch\_map\_t
- · struct touch event t

- struct menuitem\_t
- struct listmenu\_t
- struct LCDHandle

Structure defining an LCD.

struct rect t

Structure representing a rectangle.

- struct HDTEntry t
- struct HDTTable t
- struct HDTDevice t
- struct \_HDTDevCAM\_t
- struct \_HDTDevMOTOR\_t
- struct \_HDTDevENCODER\_t
- struct HDTDevSERVO t
- struct HDTDevPSD t
- struct \_HDTDevDRIVE\_t
- struct \_HDTDevIRTV\_t
- struct \_HDTDevADC\_t
- struct \_HDTDevCOM\_t

# **Typedefs**

- typedef char \* DeviceSemantics
- typedef XColor rgb\_t
- typedef unsigned short hword\_t
- typedef struct mpsse context \* SPIHandle
- typedef unsigned long keycode\_t
- typedef unsigned char keymode\_t
- · typedef unsigned short Icdmode\_t
- typedef unsigned char byte\_t
- typedef float meterPerSec
- · typedef float radPerSec
- · typedef float meter
- · typedef float radians
- typedef rect\_t menurect\_t

Structure representing a menu rectangle.

- typedef unsigned int SERVOHandle
- typedef unsigned int MOTORHandle
- · typedef unsigned int QUADHandle
- typedef unsigned int PSDHandle
- · typedef unsigned int ADCHandle
- · typedef unsigned int VWHandle
- · typedef unsigned int CAMHandle
- typedef struct \_HDTEntry\_t HDTEntry\_t
- typedef struct \_HDTTable\_t HDTTable\_t
- typedef struct HDTDevice t HDTDevice\_t

- typedef struct <u>HDTDevCAM\_t</u> HDTDevCAM\_t
- typedef struct \_HDTDevMOTOR\_t HDTDevMOTOR\_t
- typedef struct \_HDTDevENCODER\_t HDTDevENCODER\_t
- typedef struct \_HDTDevSERVO\_t HDTDevSERVO\_t
- typedef struct <u>HDTDevPSD\_t</u> HDTDevPSD\_t
- typedef struct \_HDTDevDRIVE\_t HDTDevDRIVE\_t
- typedef struct \_HDTDevIRTV\_t HDTDevIRTV\_t
- typedef struct HDTDevADC t HDTDevADC t
- typedef struct <u>HDTDevCOM\_t</u> HDTDevCOM\_t

# 5.19.1 Detailed Description

Defines types.

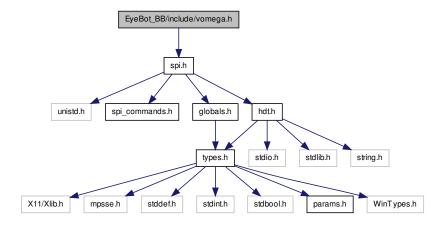
**Author** 

Remi KEAT

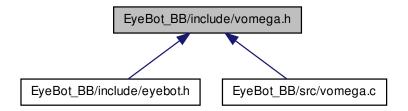
# 5.20 EyeBot\_BB/include/vomega.h File Reference

Header file for the VW functions.

#include "spi.h" Include dependency graph for vomega.h:



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



### **Functions**

• VWHandle VWInit (DeviceSemantics semantics, int Timescale)

Initialize given VW-Driver (only 1 can be initialized!). The motors and encoders are automatically reserved!! The Timescale allows to adjust the tradeoff between accuracy (scale=1, update at 100Hz) and speed(scale>1, update at 100/scale Hz).

• int VWDriveStraight (VWHandle handle, meter delta, meterPerSec v)

Drives distance "delta" with speed v straight ahead (forward or backward) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

• int VWDriveTurn (VWHandle handle, radians delta, radPerSec w)

Turns about "delta" with speed w on the spot (clockwise or counter-clockwise) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

• int VWDriveWait (VWHandle handle)

Blocks the calling process until the previous VWDriveX() command has been completed.

# 5.20.1 Detailed Description

Header file for the VW functions.

Author

Remi KEAT

# 5.20.2 Function Documentation

104 File Documentation

### 5.20.2.1 int VWDriveStraight ( VWHandle handle, meter delta, meterPerSec v )

Drives distance "delta" with speed v straight ahead (forward or backward) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

### **Parameters**

VWHandle	handle : ONE VWHandle
meter	delta : distance to drive in m
meterPer-	v : speed to drive with (always positive!)
Sec	

### delta:

- pos. -> forward
- neg. -> backward

### Returns

int retVal:

0 = ok

-1 = error wrong handle

# 5.20.2.2 int VWDriveTurn ( VWHandle handle, radians delta, radPerSec w )

Turns about "delta" with speed w on the spot (clockwise or counter-clockwise) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

### **Parameters**

VWHandle	handle : ONE VWHandle
radians	delta : degree to turn in radians
radPerSec	w : speed to turn with (always positive!)

### delta:

- pos. -> counter-clockwise
- · neg. -> clockwise

### Returns

int retVal:

0 = ok

-1 = error wrong handle

### 5.20.2.3 int VWDriveWait ( VWHandle handle )

Blocks the calling process until the previous VWDriveX() command has been completed.

### **Parameters**

VWHandle	handle : ONE VWHandle

### Returns

# int retVal:

- -1 = error wrong handle
- 0 = previous VWDriveX command has been completed

# 5.20.2.4 VWHandle VWInit ( DeviceSemantics semantics, int Timescale )

Initialize given VW-Driver (only 1 can be initialized!). The motors and encoders are automatically reserved!! The Timescale allows to adjust the tradeoff between accuracy (scale=1, update at 100Hz) and speed(scale>1, update at 100/scale Hz).

### **Parameters**

Device-	semantics
Semantics	
int	Timescale : prescale value for 100Hz IRQ (1 to)

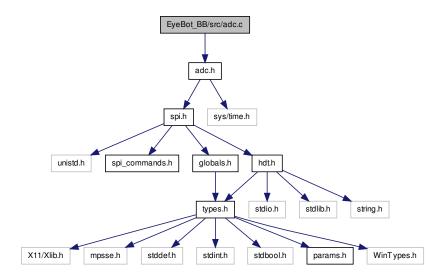
### Returns

VWHandle handle: VWHandle or 0 for error

# 5.21 EyeBot\_BB/src/adc.c File Reference

Defines the ADC functions.

#include "adc.h" Include dependency graph for adc.c:



# **Functions**

- ADCHandle OSInitADC (DeviceSemantics semantics)
  - Captures one single 10bit value from the specified adc channel.
- int OSGetADC (ADCHandle adchandle)

Captures one single 10bit value from specified AD-channel. The return value is stored in the least significant bits of the 32 bit return value.

- int ConvADCSampleToVoltage (ADCHandle adchandle, char \*volt, int sample)
  - Convert the adc sample to voltage.
- int OSADCRelease (ADCHandle handle)

Release the adc channel.

# 5.21.1 Detailed Description

Defines the ADC functions.

Author

Remi KEAT

# 5.21.2 Function Documentation

### 5.21.2.1 int ConvADCSampleToVoltage ( ADCHandle adchandle, char \* volt, int sample )

Convert the adc sample to voltage.

# **Parameters**

ADCHandle	adchandle : desired AD-channel
char*	volt : pointer to string
int	sample : ADC sample

Result is stored in char \*volt. Valid values: ADC0, ADC1, ADC2, ADC3

### Returns

int retVal : 0: ok -1: invalid channel

# 5.21.2.2 int OSADCRelease ( ADCHandle handle )

Release the adc channel.

### **Parameters**

ADCHandle	handle

### Returns

int retVal : always 0

# 5.21.2.3 int OSGetADC ( ADCHandle adchandle )

Captures one single 10bit value from specified AD-channel. The return value is stored in the least significant bits of the 32 bit return value.

### **Parameters**

ADCHandle   handle : Handler for the adc channel
--

### **Returns**

int retVal >0: 10 bit sampled value

-1: invalid channel

# 5.21.2.4 ADCHandle OSInitADC ( DeviceSemantics semantics )

Captures one single 10bit value from the specified adc channel.

### **Parameters**

Device-	semantics : desired ADC channel
Semantics	

# Returns

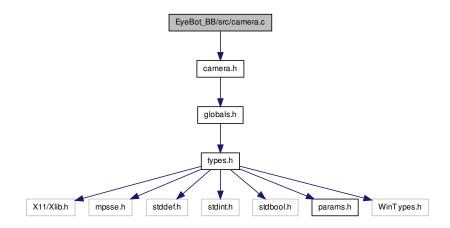
ADCHandle handle >0: Handler for the adc channel 0: Initialization error

Valid values for semantics: ADC0, ADC1, ADC2, ADC3

# 5.22 EyeBot\_BB/src/camera.c File Reference

Defines functions for the camera.

#include "camera.h" Include dependency graph for camera.c:



# **Functions**

- CAMHandle CAMInit (DeviceSemantics semantics)
  - Configure & Initialize camera.
- int CAMGetFrameRGB (CAMHandle handle, BYTE \*buf)

Reads one full color image in RBG format, 3 bytes per pixel.

# 5.22.1 Detailed Description

Defines functions for the camera.

Author

Remi KEAT

# 5.22.2 Function Documentation

5.22.2.1 int CAMGetFrameRGB ( CAMHandle handle, BYTE \* buf )

Reads one full color image in RBG format, 3 bytes per pixel.

### **Parameters**

CAMHandle	handle : handle of the desired camera
BYTE*	buf : pointer to image buffer of full size (use CAMGet)

# **Returns**

int retVal : return code

0 = success

-1 = error (camera not initialized)

# 5.22.2.2 CAMHandle CAMInit ( DeviceSemantics semantics )

Configure & Initialize camera.

# **Parameters**

Device-	semantics : handle of the desired camera
Semantics	

### Returns

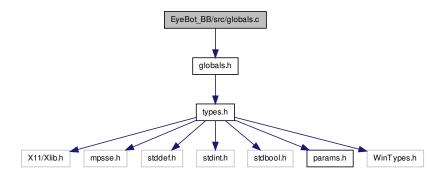
CAMHandle handle

# 5.23 EyeBot\_BB/src/globals.c File Reference

Defines global variables.

110 File Documentation

#include "globals.h" Include dependency graph for globals.c:



# **Variables**

- struct mpsse\_context \* gDeviceHandle
- LCDHandle \* gLCDHandle
- bool gLCDEnabled
- int gCurPosX
- int gCurPosY
- int gMousePosX
- int gMousePosY
- int gMouseButton
- touch\_map\_t \* gTouchMap

# 5.23.1 Detailed Description

Defines global variables.

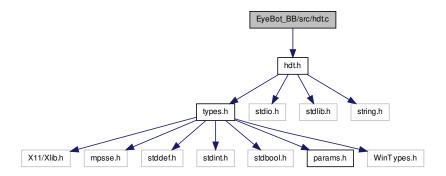
Author

Remi KEAT

# 5.24 EyeBot\_BB/src/hdt.c File Reference

Defines functions used by the HDT system.

#include "hdt.h" Include dependency graph for hdt.c:



# **Classes**

struct HDTTypes t

# **Typedefs**

• typedef struct \_HDTTypes\_t HDTTypes\_t

# **Functions**

- \_\_ssize\_t getline (char \*\*\_\_restrict \_\_lineptr, size\_t \*\_\_restrict \_\_n, FILE \*\_\_restrict \_\_stream)
- int HDTValidate (char \*filename)

checks all HDT entries in given filename. will not check for specific entry (only check entry headers).

- int HDTListEntry (char \*filename, HDTEntry t \*deventry, int count)
  - Copy all entries to deventry. user need to free the allocated memory by using free(deventry->buffer). return value may be less than count.
- int HDTFindEntry (void \*hdtfile, char \*devname, HDTEntry\_t \*deventry)
  - finds an entry in the hdt file that matches given name and copies the entry to given structure. the newline character is replaced by null. user need to free the allocated memory by using free(deventry->buffer).
- int HDTFindTable (void \*hdtfile, char \*tabname, HDTTable\_t \*tabentry)

finds a table in the hdt file that matches given name and copies the table data to given structure.

- HDTTable\_t \* HDTLoadTable (char \*filename, HDTDevice\_t \*pdevices)
  - load all tables needed by pdevices if found. the return value is a pointer to the first table. the tables are in a linked list allocated with dynamic memory. use HDTClear-Table to free up the resources.

int HDTClearTable (HDTTable t \*ptables)

Free the allocated resources for the tables created by HDTLoadTable.

HDTDevCAM t \* HDTLoadCAM (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearCAM (HDTDevCAM t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevMOTOR t \* HDTLoadMOTOR (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearMOTOR (HDTDevMOTOR t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevENCODER t \* HDTLoadENCODER (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearENCODER (HDTDevENCODER\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

 int HDTLinkENC2MOT (HDTDevENCODER\_t \*pencoders, HDTDevMOTOR\_t \*pmotors)

Link the encoders to the motors.

• HDTDevPSD\_t \* HDTLoadPSD (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearPSD (HDTDevPSD\_t \*pdevs)

Free the allocated resources for the < device> created by HDTLoad< device>.

• HDTDevSERVO\_t \* HDTLoadSERVO (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearSERVO (HDTDevSERVO t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevDRIVE\_t \* HDTLoadDRIVE (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearDRIVE (HDTDevDRIVE t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

int HDTLinkDRV2ENC (HDTDevDRIVE\_t \*pdrives, HDTDevENCODER\_t \*pencoders)

Link the drives to the encoders.

• HDTDevIRTV t \* HDTLoadIRTV (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearIRTV (HDTDevIRTV\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

HDTDevADC t \* HDTLoadADC (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearADC (HDTDevADC\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

• HDTDevCOM t \* HDTLoadCOM (char \*filename, char \*devname)

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

int HDTClearCOM (HDTDevCOM\_t \*pdevs)

Free the allocated resources for the <device> created by HDTLoad<device>.

# 5.24.1 Detailed Description

Defines functions used by the HDT system.

Author

Remi KEAT

# 5.24.2 Function Documentation

```
5.24.2.1 __ssize_t getline ( char **__restrict __lineptr, size_t *__restrict __n, FILE *__restrict __stream )
```

20080917 - azman@ee.uwa.edu.au

# 5.24.2.2 int HDTClearADC ( HDTDevADC t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

### **Parameters**

HDTDovAD	pdevs : <device> list to be cleared</device>
no i bevab-	puevs . < device > list to be cleared
<i>Ct</i> *_	

Generated on Sat Sep 14 2013 08:41:42 for EyeBot\_BB by Doxygen

int retVal: always 0

# 5.24.2.3 int HDTClearCAM ( HDTDevCAM\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

### **Parameters**

HDTDevCA-	pdevs : <device> list to be cleared</device>
<i>M_t</i> *	

# Returns

int retVal: always 0

# 5.24.2.4 int HDTClearCOM ( HDTDevCOM\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

### **Parameters**

HDTDevCO-	pdevs : <device> list to be cleared</device>
<i>M_t</i> ∗	

### Returns

int retVal: always 0

# 5.24.2.5 int HDTClearDRIVE ( HDTDevDRIVE\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

# **Parameters**

HDTDevDR-	pdevs : <device> list to be cleared</device>
IVE_t*	

# Returns

int retVal: always 0

# $5.24.2.6 \quad \text{int HDTClearENCODER ( HDTDevENCODER\_t} * \textit{pdevs })$

Free the allocated resources for the <device> created by HDTLoad<device>.

# **Parameters**

HDTDevEN-	pdevs : <device> list to be cleared</device>
CODER_t*	

### **Returns**

int retVal: always 0

# 5.24.2.7 int HDTClearIRTV ( HDTDevIRTV\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

# **Parameters**

HDTDevIRT-	pdevs : <device> list to be cleared</device>
<i>V_t</i> ∗	

### **Returns**

int retVal: always 0

# $5.24.2.8 \quad \text{int HDTClearMOTOR} \left( \ \text{HDTDevMOTOR} \underline{} t * \textit{pdevs} \ \right)$

Free the allocated resources for the <device> created by HDTLoad<device>.

### **Parameters**

Ī	HDTDevMO-	pdevs : <device> list to be cleared</device>
	TOR_t*	

### Returns

int retVal: always 0

# 5.24.2.9 int HDTClearPSD ( HDTDevPSD\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

### **Parameters**

HDTDevPS-	pdevs : <device> list to be cleared</device>
D_t*	

int retVal: always 0

# 5.24.2.10 int HDTClearSERVO ( HDTDevSERVO\_t \* pdevs )

Free the allocated resources for the <device> created by HDTLoad<device>.

### **Parameters**

HDTDevSE-	pdevs : <device> list to be cleared</device>
RVO_t*	

# Returns

int retVal: always 0

# 5.24.2.11 int HDTClearTable ( HDTTable\_t \* ptables )

Free the allocated resources for the tables created by HDTLoadTable.

### **Parameters**

HDTTable	ptables : tables to be cleared
t*	

### Returns

int retVal: always 0

# 5.24.2.12 int HDTFindEntry (void \* hdtfile, char \* devname, HDTEntry\_t \* deventry)

finds an entry in the hdt file that matches given name and copies the entry to given structure. the newline character is replaced by null. user need to free the allocated memory by using free(deventry->buffer).

### **Parameters**

void*	hdtfile : hdt file fopen with "rt" flag
char*	devname : name of entry to search for
HDTEntry	deventry: storage structure for the entry
t*	

int retVal:

-1 on failure (no entry found) [entry length] on success

5.24.2.13 int HDTFindTable ( void \* hdtfile, char \* tabname, HDTTable\_t \* tabentry )

finds a table in the hdt file that matches given name and copies the table data to given structure.

### **Parameters**

void*	hdtfile : hdt file (fopen with "rt" flag)
char*	tabname : name of table to search for
HDTTable	tabentry: storage structure for the table
t*	

### **Returns**

int retVal:

-1 on failure (no table found) [table size] on success

# 5.24.2.14 int HDTLinkDRV2ENC ( HDTDevDRIVE\_t \* pdrives, HDTDevENCODER\_t \* pencoders )

Link the drives to the encoders.

# **Parameters**

	HDTDevDR-	pdrives : list of drive methods
	IVE_t*	
Ī	HDTDevEN-	pencoders : list of encoders
	$CODER_t*$	

### **Returns**

int retVal:

0 on success

Negative value on failure (number of unconnected link)

# 5.24.2.15 int HDTLinkENC2MOT ( HDTDevENCODER\_t \* pencoders, HDTDevMOTOR\_t \* pmotors )

Link the encoders to the motors.

### **Parameters**

HDTDevE	N- pencoders : list of encoders
CODER	_t*
HDTDevM	O- pmotors : list of motors
TOR	_t*

### Returns

int retVal:

0 on success

Negative value on failure (number of unconnected link)

5.24.2.16 int HDTListEntry ( char \* filename, HDTEntry\_t \* deventry, int count )

Copy all entries to deventry. user need to free the allocated memory by using free(deventry->buffer). return value may be less than count.

### **Parameters**

char*	filename : name of HDT file to be checked for listing
HDTEntry	deventry : storage structure for the entry
t*	
int	count : number of deventry storage supplied

### Returns

int retVal:

-1 on failure

(number of entries) on success

 $\textbf{5.24.2.17} \quad \textbf{HDTDevADC\_t}*\ \textbf{HDTLoadADC}\ (\ \textbf{char}*\ \textit{filename},\ \textbf{char}*\ \textit{devname}\ )$ 

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

# **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

### Returns

HDTDevADC\_t\* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found

### 5.24.2.18 HDTDevCAM\_t\* HDTLoadCAM ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

### **Returns**

```
HDTDevCAM_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

# $\textbf{5.24.2.19} \quad \textbf{HDTDevCOM\_}t* \, \textbf{HDTLoadCOM} \, ( \, \, \textbf{char} * \textit{filename}, \, \, \textbf{char} * \textit{devname} \, \, )$

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

### Parameters

char*	filename : hdt file to open
char	*devname : device semantics

### **Returns**

```
HDTDevCOM_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

# 5.24.2.20 HDTDevDRIVE\_t\* HDTLoadDRIVE ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

120 File Documentation

### Returns

```
HDTDevDRIVE_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

```
5.24.2.21 HDTDevENCODER_t* HDTLoadENCODER ( char * filename, char * devname )
```

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

#### Returns

```
HDTDevENCODER_t* handle : 0x0 on failure (no <device> found) (pointer to first <device>) if found
```

```
5.24.2.22 HDTDevIRTV_t* HDTLoadIRTV ( char * filename, char * devname )
```

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

### Returns

```
HDTDevIRTV_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

```
5.24.2.23 HDTDevMOTOR_t* HDTLoadMOTOR ( char * filename, char * devname )
```

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <devices>

are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

#### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

### Returns

```
HDTDevMOTOR_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

# 5.24.2.24 HDTDevPSD\_t\* HDTLoadPSD ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

### **Parameters**

ĺ	char*	filename : hdt file to open
I	char*	devname : device semantics

### **Returns**

```
HDTDevIRTV_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

### 5.24.2.25 HDTDevSERVO\_t\* HDTLoadSERVO ( char \* filename, char \* devname )

load all <device> entry found in the hdt file if devname is null. else, load only the requested device. the return value is a pointer to the first <device>. the <device> are in a linked list allocated with dynamic memory. use HDTClear<device> to free up the resources.

### **Parameters**

char*	filename : hdt file to open
char*	devname : device semantics

File Documentation

# Returns

122

```
HDTDevSERVO_t* handle :
0x0 on failure (no <device> found)
(pointer to first <device>) if found
```

```
5.24.2.26 \quad \textbf{HDTTable\_t}*\ \textbf{HDTLoadTable}\ (\ \textbf{char}*\ \textit{filename},\ \textbf{HDTDevice\_t}*\ \textit{pdevices}\ )
```

load all tables needed by pdevices - if found. the return value is a pointer to the first table. the tables are in a linked list allocated with dynamic memory. use HDTClearTable to free up the resources.

### **Parameters**

char*	filename : hdt file to open
HDTDevice-	pdevices : devices with tablename in linked list
_ <i>t</i> *	

### Returns

```
HDTTable_t* table :
0x0 on failure (no table found)
(pointer to first table) if found
```

# 5.24.2.27 int HDTValidate ( char \* filename )

checks all HDT entries in given filename. will not check for specific entry (only check entry headers).

### **Parameters**

char*	filename: name of HDT file to be checked
-------	--

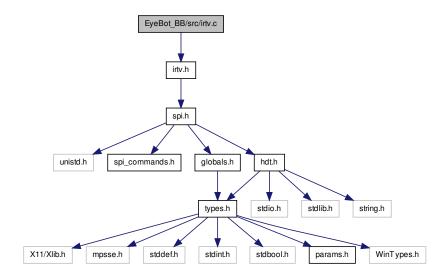
int retVal:

-1 if incorrect HDT entry found (number of entries) if otherwise

# 5.25 EyeBot\_BB/src/irtv.c File Reference

### Defines IRTV functions.

#include "irtv.h" Include dependency graph for irtv.c:



### **Functions**

- int IRTVInit (DeviceSemantics semantics)
   Initializes the IR remote control decoder by calling IRTVInit() with the device name
- int IRTVRead (void)

Reads and removes the next key code from the code buffer. Does not wait.

• void IRTVRelease (void)

Terminates the remote control decoder and releases the irtv thread.

# 5.25.1 Detailed Description

Defines IRTV functions.

found in the corresponding HDT entry.

# Author

Remi KEAT

# 5.25.2 Function Documentation

# 5.25.2.1 int IRTVInit ( DeviceSemantics semantics )

Initializes the IR remote control decoder by calling IRTVInit() with the device name found in the corresponding HDT entry.

### **Parameters**

Devic	e- semantics
Semantio	es

### Returns

int retVal:

- 0 = ok
- 1 = HDT file error
- 2 = invalid or missing "IRTV" HDT entry for this semantics

# 5.25.2.2 int IRTVRead (void)

Reads and removes the next key code from the code buffer. Does not wait.

# Returns

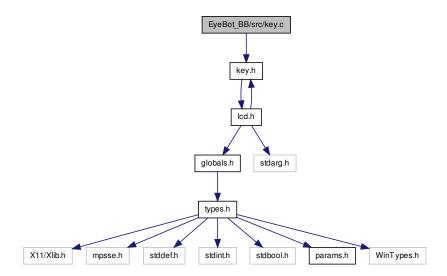
int retVal: Next code from the buffer

0 = no key

# 5.26 EyeBot\_BB/src/key.c File Reference

Defines functions for the key input.

#include "key.h" Include dependency graph for key.c:



# **Functions**

• int KEYInit (void)

Open the evdev device file for reading touch events. Load the key configuration file (if found), else use the hardcoded default value.

• keymode\_t KEYSetTM (keymode\_t mode)

Set mode for key touch map.

keymode\_t KEYGetTM (touch\_map\_t \*\*ptouch\_map)

Get current mode and touch map (region map).

• int KEYSetRegion (int index, m6key\_box\_t \*region)

Manually set region data into current touch map. Only used in KEYTM\_REGIONS mode. If region is 0x0, resets the touch map (mode becomes KEYTM\_UNKNOWN).

• int KEYGetRegion (int index, m6key\_box\_t \*region)

Copy specific region data out from the current touch map. Only used in KEYTM\_RE-GIONS mode.

int KEYNoTouch (touch event t \*rawtouch)

Validate there's no touch on screen surface.

int KEYGetRAW (touch\_event\_t \*rawtouch)

Gets raw touch info - a non-blocking function. Mainly used for calibration and testing.

keycode\_t KEYDecode (touch\_event\_t \*rawtouch)

Decode raw touch info into a keycode based on the current touch map. Mainly used for testing.

· int KEYRelease (void)

126 File Documentation

Close the evdev device file and stop checking any key touch event.

• int inside (int x, int y, m6key\_box\_t rect)

Check if a point is inside a rectangle.

• int KEYIdle (int idle)

Enable/disable event checking procedure.

keycode\_t KEYWait (keycode\_t excode)

Wait for specific keys only.

keycode\_t KEYGet (void)

Wait for a touch event and return keycode (including KEY\_INVALID - undefined keycode).

coord\_pair\_t KEYGetXY (void)

Wait for a touch event and return the XY-coordinate.

keycode\_t KEYRead (void)

Read a keycode and returns. Function does not wait, thus includes KEY\_TIMEOUT.

• int activate\_escape (int escape)

# 5.26.1 Detailed Description

Defines functions for the key input.

**Author** 

Remi KEAT

# 5.26.2 Function Documentation

5.26.2.1 int inside ( int x, int y, m6key\_box\_t rect )

Check if a point is inside a rectangle.

### **Parameters**

int	x : X-coordinate of the point
int	y: Y-coordinate of the point
m6key_box-	rect : rectangle structure
_t	

### Returns

int retVal: non-null if inside

# $\textbf{5.26.2.2} \quad \textbf{keycode\_t KEYDecode} \left( \ \textbf{touch\_event\_t} * \textit{rawtouch} \ \right)$

Decode raw touch info into a keycode based on the current touch map. Mainly used for testing.

# **Parameters**

touch_event-	rawtouch : pointer to touch_event_t structure
_t*	

### **Returns**

keycode\_t keyCode : Status of touch data (variable in rawtouch)

# 5.26.2.3 keycode\_t KEYGet ( void )

Wait for a touch event and return keycode (including KEY\_INVALID - undefined keycode).

### **Returns**

keycode\_t retKey : Keycode value

# 5.26.2.4 int KEYGetRAW ( $touch_event_t * rawtouch$ )

Gets raw touch info - a non-blocking function. Mainly used for calibration and testing.

# Parameters

touch_event-	rawtouch : pointer to touch_event_t structure
_ <i>t</i> *	

# Returns

int retVal:

0 if sync signal received!

Negative value if otherwise

# 5.26.2.5 int KEYGetRegion (int index, m6key\_box\_t \* region)

Copy specific region data out from the current touch map. Only used in KEYTM\_REGIONS mode.

### **Parameters**

int	index : Index for region
m6key_box-	region : Pointer to a storage for region data
_t*	

int retVal: 0 on success Negative value on failure

```
5.26.2.6 keymode_t KEYGetTM ( touch_map_t ** ptouch_map )
```

Get current mode and touch map (region map).

### **Parameters**

```
touch_map- ptouch_map : Pointer to a touch_map_t structure
_t**
```

### Returns

keymode\_t retMod : Current touch map mode

```
5.26.2.7 coord_pair_t KEYGetXY ( void )
```

Wait for a touch event and return the XY-coordinate.

### Returns

```
coord_pair_t retCoord : Coordinate pair
```

# 5.26.2.8 int KEYIdle (int idle)

Enable/disable event checking procedure.

# Parameters

```
int idle : user request
```

Valid values for idle:

- KEY\_GOIDLE deactivate event checking
- · KEY\_NOIDLE activate event checking
- KEY\_STATE request current status

### Returns

int status: Idle status of event checking procedure

```
5.26.2.9 int KEYInit (void)
```

Open the evdev device file for reading touch events. Load the key configuration file (if found), else use the hardcoded default value.

#### Returns

int retVal : 0 on success Negative value on failure

```
5.26.2.10 int KEYNoTouch ( touch_event_t * rawtouch )
```

Validate there's no touch on screen surface.

#### **Parameters**

touch_event-	rawtouch: pointer to touch_event_t structure this is optional! only if raw
_ <i>t</i> *	data needed! else, use 0x0!

#### **Returns**

int retVal:

0 - being touched

1 - not touched

```
5.26.2.11 keycode_t KEYRead ( void )
```

Read a keycode and returns. Function does not wait, thus includes KEY\_TIMEOUT.

### Returns

```
keycode_t retKey : Keycode value
```

```
5.26.2.12 int KEYRelease (void)
```

Close the evdev device file and stop checking any key touch event.

#### Returns

```
int retVal: 0 on success
Negative value on failure
```

```
5.26.2.13 int KEYSetRegion ( int index, m6key\_box\_t * region )
```

Manually set region data into current touch map. Only used in KEYTM\_REGIONS mode. If region is 0x0, resets the touch map (mode becomes KEYTM\_UNKNOWN).

### **Parameters**

int	index : Index for region
m6key_box-	region : Pointer to a region data
_t*	

### Returns

int retVal: 0 on success Negative value on failure

5.26.2.14 keymode\_t KEYSetTM ( keymode\_t mode )

Set mode for key touch map.

### **Parameters**

keymode_t	mode : Requested touch map mode
-----------	---------------------------------

## Returns

keymode\_t retMod : Current touch map mode

5.26.2.15 keycode\_t KEYWait ( keycode\_t excode )

Wait for specific keys only.

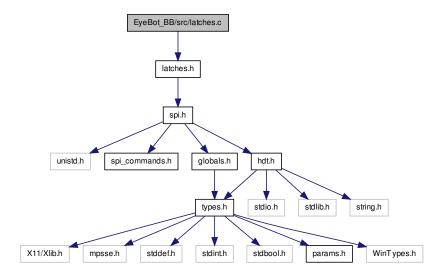
KAVCAMA T	excode : Expected keycode values (bit XORed)
NEYCOUE L	EXCOURT LADECIEU REVOOUE VAIUES (DIL ACTIEU)
,	choose i =hpooled hojoode raidoo (ali rio iloa)

keycode\_t retKey : Keycode value

# 5.27 EyeBot\_BB/src/latches.c File Reference

Defines functions to control latches.

#include "latches.h" Include dependency graph for latches.c:



## **Functions**

• int OSLatchSetup (int latchnum, int direction)

Setup the given latch as input or output.

• int OSLatchBankSetup (int banknum, int direction)

Setup the given io buffer bank as input or output.

• int OSLatchRead (int latchnum)

Read content of the selected input latch.

• int OSLatchWrite (int latchnum, int state)

Write to the selected output latch.

• int OSLatchInit (void)

Initialize the digital IO, call this before using any digital IO functions.

• int OSLatchCleanup (void)

Unmap the memory for digital IOs, call these when the digital IOs functions are no longer needed.

## 5.27.1 Detailed Description

Defines functions to control latches.

**Author** 

Remi KEAT

### 5.27.2 Function Documentation

## 5.27.2.1 int OSLatchBankSetup ( int banknum, int direction )

Setup the given io buffer bank as input or output.

### **Parameters**

int	banknum : bank number
int	direction : signal direction

Valid values for direction:

- 0 = input
- 1 = output

### Note:

- LATCH0..LATCH7 are connected to IOBANK0
- · LATCH8..LATCH15 are connected to IOBANK1

### Returns

int retVal: always 0

## 5.27.2.2 int OSLatchCleanup (void)

Unmap the memory for digital IOs, call these when the digital IOs functions are no longer needed.

### Returns

int retVal: always 0

### 5.27.2.3 int OSLatchInit (void)

Initialize the digital IO, call this before using any digital IO functions.

int retVal

Return code:

- 0 = ok
- -1 = Initialization error

## 5.27.2.4 int OSLatchRead (int latchnum)

Read content of the selected input latch.

## **Parameters**

int	latchnum : latch number to read

Return latch status:

- 0 = low
- 1 = high

### Returns

int readValue

## 5.27.2.5 int OSLatchSetup ( int latchnum, int direction )

Setup the given latch as input or output.

### **Parameters**

int	latchnum : latch number
int	direction : signal direction

Valid values for direction:

- 0 = input
- 1 = output

## Returns

int retVal: always 0

### 5.27.2.6 int OSLatchWrite (int latchnum, int state)

Write to the selected output latch.

### **Parameters**

int	latchnum : latch number to write
int	state : state to be set to the selected out latch

### Valid values for state:

- 0 = low
- 1 = high

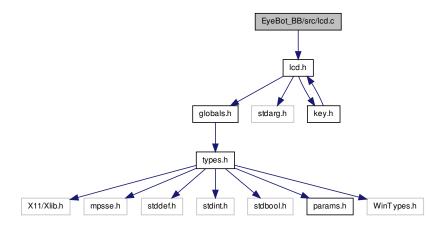
#### Returns

int retVal : always 0

# 5.28 EyeBot\_BB/src/lcd.c File Reference

Defines functions to interact with the LCD screen.

#include "lcd.h" Include dependency graph for lcd.c:



## **Functions**

• rgb\_t getColor (char \*colorName)

Return the rgb\_t color from the color name.

int LCDDrawFrame (int x1, int y1, int x2, int y2, rgb\_t color)

Draw a bordered frame.

• int LCDDrawMenu (void)

Draw the menu.

int LCDDrawList (void)

Draw the list.

• int LCDInit ()

Initialize the LCD.

int LCDClear (void)

Clear the LCD display and all display buffers.

int LCDSetMode (hword t mode)

Update the internal mode flag bits.

hword\_t LCDGetMode (void)

Get the internal mode flag bits.

int LCDResetMode (hword t mode)

Reset the internal mode flag bits to a previously saved mode.

• int LCDMenu (char \*string1, char \*string2, char \*string3, char \*string4)

Set menu entries in KEY\_CLASSIC mode (4-buttons). Also sets the LCD\_SHOWME-NU flag and refresh the LCD.

• int LCDMenul (int pos, char \*string, rgb\_t fgcol, rgb\_t bgcol, void \*userp)

Set specific menu entry in KEY\_CLASSIC mode (index given by pos). Color customization for specific key is now possible (fgcol/bgcol). A user-specific data can be linked to the menu using userp pointer. Will also set the LCD\_SHOWMENU flag and refresh the LCD.

• menuitem\_t \* LCDMenuItem (int index)

Return the menuitem at a given position.

int LCDList (listmenu\_t \*menulist)

Setup the list menu display and update appropriate info in the listmenu\_t structure pointed by menulist (e.g. scroll, count). Will also set the LCD\_LISTMENU flag and refresh the LCD.

• int LCDSetList (listmenu t \*menulist)

Unlike LCDList(), this will blindly assign menulist to the mainlist for display. Doesn't update anything in the menulist structure, nor modify any internal flags. Useful to maintain multiple lists fo menu display.

• listmenu t \* LCDGetList (void)

Get the currently active list menu.

menurect t \* LCDListBox (int pos)

Get the frame info of a specific list item in form of a menurect\_t structure.

• menuitem\_t \* LCDListActiveItem (void)

Get the selected menuitem in the list menu – using index & start variable in listmenu\_t. Will return 0x0 (NUL) if no item is currently selected.

int LCDArea (int x1, int y1, int x2, int y2, rgb\_t color)

Draw a color-filled rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

• int LCDFrame (int x1, int y1, int x2, int y2, rgb t color)

136 File Documentation

Draw a color rectangle frame with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

• int LCDTextColor (rgb\_t fgcol, rgb\_t bgcol, char colorflags)

Set the default color for text (including background) and related flags (e.g. for transparent background).

• int LCDPrintf (const char \*format,...)

Print formatted string to LCD and refresh LCD. Cursor position is updated.

• int LCDSetPrintf (int row, int column, const char \*format,...)

LCDPrintf with text position specified.

• int LCDPutChar (char c)

Write a character to LCD and refresh LCD. Cursor position is updated.

• int LCDSetChar (int row, int column, char c)

LCDPutChar with text position specified.

int LCDPutString (char \*string)

Print string to LCD and refresh LCD. Cursor position is updated.

int LCDSetString (int row, int column, char \*string)

LCDPutString with text position specified.

int LCDPutHex (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

int LCDPutHex1 (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

• int LCDPutInt (int val)

Print integer to LCD and refresh LCD. Cursor position is updated.

int LCDPutIntS (int val, int spaces)

Print integer to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

• int LCDPutFloat (float val)

Print floating-point value to LCD and refresh LCD. Cursor position is updated.

• int LCDPutFloatS (float val, int spaces, int decimals)

Print floating-point value to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

• int LCDSetPos (int row, int column)

Set the text cursor position to (row, column).

int LCDGetPos (int \*row, int \*column)

Get the text cursor position.

int LCDSetPixel (int x, int y, rgb\_t color)

Sets the color of the pixel at (x,y) coordinate to color.

rgb\_t LCDGetPixel (int x, int y)

Get the RGB color value of the pixel at (x,y) coordinate.

rgb\_t InvertColor (rgb\_t color)

Invert a RGB color.

• int LCDInvertPixel (int x, int y)

Bit-invert the color of the pixel at (x,y) coordinate.

int LCDLine (int x1, int y1, int x2, int y2, rgb\_t color)

Draw a color line from (x1,y1) to (x2,y2).

• int LCDLineInvert (int x1, int y1, int x2, int y2)

Draw a line from (x1,y1) to (x2,y2). The line pixels will invert the color of existing pixels.

int LCDAreaInvert (int x1, int y1, int x2, int y2)

Draw a rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate. The pixels in the specified area will invert the color of existing pixels.

• rect t LCDTextBar (int row, int column, int length, int fill, rgb t color)

Draw a textbar for text starting at position (row, column) until (row, column+length). The textbar will take about 25%-50% of text height & width to draw its frame. The fill parameter will define how much of the text bar should be 'filled' with color (like a progress bar).

• int LCDNeedRefresh (void)

Indicate if the LCD need to be refreshed.

• int LCDRelease ()

Release the LCD.

• int LCDRefresh (void)

Refresh the screen (i.e write display buffers to the framebuffer device).

int LCDGetFBInfo (fbinfo t \*pinfo)

Get display information and save to structure pointed by pinfo. Cursor info needs LC-Dlnit() for textsize.

• int LCDListCount (void)

Get the number of list items supported by the current display (text) configuration. - This includes the item for title bar - thus, different from count variable in listmenu\_t as updated by an LCDList() call.

• int LCDListIndex (int index)

Set the list index.

int LCDListScrollUp (void)

Scrolls the list display up. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

• int LCDListScrollDown (void)

Scrolls the list display down. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

int LCDPutImageRGB (int xpos, int ypos, int xsize, int ysize, byte\_t \*data)

Place a RGB color image (24bpp) at (xpos,ypos) position on the LCD screen.

## 5.28.1 Detailed Description

Defines functions to interact with the LCD screen.

Author

Remi KEAT

### 5.28.2 Function Documentation

```
5.28.2.1 rgb_t getColor ( char * colorName )
```

Return the rgb\_t color from the color name.

### **Parameters**

char*	colorName
-------	-----------

## Returns

rgb\_t color

## 5.28.2.2 rgb\_t InvertColor ( rgb\_t color )

Invert a RGB color.

### **Parameters**

rgb_t color : RGB color value
-------------------------------

### Returns

rgb t color: RGB color value

## 5.28.2.3 int LCDArea ( int x1, int y1, int x2, int y2, $rgb_t color$ )

Draw a color-filled rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel
rgb t	color : RGB fill color value

int retVal: always 0

## 5.28.2.4 int LCDAreaInvert (int x1, int y1, int x2, int y2)

Draw a rectangle with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate. The pixels in the specified area will invert the color of existing pixels.

### **Parameters**

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel

#### **Returns**

int retVal: always 0

5.28.2.5 int LCDClear (void)

Clear the LCD display and all display buffers.

### Returns

int retVal: always 0

5.28.2.6 int LCDDrawFrame ( int x1, int y1, int x2, int y2, rgb\_t color )

Draw a bordered frame.

### **Parameters**

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel

### Returns

int retVal: always 0

5.28.2.7 int LCDDrawList (void)

Draw the list.

int retVal : always 0

## 5.28.2.8 int LCDDrawMenu (void)

Draw the menu.

### Returns

int retVal : always 0

## 5.28.2.9 int LCDFrame ( int x1, int y1, int x2, int y2, $rgb_t color$ )

Draw a color rectangle frame with (x1,y1) as top-left coordinate and (x2,y2) as the bottom-right coordinate.

#### **Parameters**

int	x1 : X-coordinate of top-left pixel
int	y1 : Y-coordinate of top-left pixel
int	x2 : X-coordinate of bottom-right pixel
int	y2 : Y-coordinate of bottom-right pixel
rgb_t	color : RGB fill color value

### Returns

int retVal: always 0

## 5.28.2.10 int LCDGetFBInfo (fbinfo\_t \* pinfo)

Get display information and save to structure pointed by pinfo. Cursor info needs LCD-Init() for textsize.

## Parameters

fh	info t*	ninfo :	: Pointer to storage for screen & cursor info	
.~		P	. I dilitar to didrago for corderi a dareer inic	

### Returns

int retVal

0 on success

Negative value on failure

```
5.28.2.11 listmenu_t* LCDGetList ( void )
```

Get the currently active list menu.

### **Returns**

```
listmenu_t* retListMenu : Pointer to listmenu_t structure
```

```
5.28.2.12 hword_t LCDGetMode (void)
```

Get the internal mode flag bits.

#### **Returns**

hword\_t mode : Current mode flag bits

```
5.28.2.13 rgb_t LCDGetPixel ( int x, int y )
```

Get the RGB color value of the pixel at (x,y) coordinate.

### **Parameters**

int	x : X-coordinate of the pixel
int	y: Y-coordinate of the pixel

### Returns

```
rgb_t color : RGB color value
```

```
5.28.2.14 int LCDGetPos ( int * row, int * column )
```

Get the text cursor position.

### **Parameters**

int*	row : Pointer to cursor row index
<i>int</i> *	column : Pointer to cursor column index

### **Returns**

int retVal: always 0

5.28.2.15 int LCDInit ( )

Initialize the LCD.

int retVal: always 0

## 5.28.2.16 int LCDInvertPixel (int x, int y)

Bit-invert the color of the pixel at (x,y) coordinate.

#### **Parameters**

int	x : X-coordinate of the pixel
int	y: Y-coordinate of the pixel

## Returns

int retVal: always 0

## 5.28.2.17 int LCDLine ( int x1, int y1, int x2, int y2, $rgb_t color$ )

Draw a color line from (x1,y1) to (x2,y2).

### **Parameters**

int	x1 : X-coordinate of first pixel
int	y1 : Y-coordinate of first pixel
int	x2 : X-coordinate of second pixel
int	y2 : Y-coordinate of second pixel
rgb_t	color : RGB color value for the pixel

## Returns

int retVal : always 0

## 5.28.2.18 int LCDLineInvert (int x1, int y1, int x2, int y2)

Draw a line from (x1,y1) to (x2,y2). The line pixels will invert the color of existing pixels.

	int	x1 : X-coordinate of first pixel
ĺ	int	y1 : Y-coordinate of first pixel
ĺ	int	x2 : X-coordinate of second pixel
ĺ	int	y2 : Y-coordinate of second pixel

int retVal: always 0

5.28.2.19 int LCDList ( listmenu\_t \* menulist )

Setup the list menu display and update appropriate info in the <a href="listmenu\_t">listmenu\_t</a> structure pointed by menulist (e.g. scroll, count). Will also set the LCD\_LISTMENU flag and refresh the LCD.

#### **Parameters**

```
listmenu_t* menulist : Listmenu to be used for display
```

### **Returns**

int retVal: always 0

5.28.2.20 menuitem\_t\* LCDListActiveItem ( void )

Get the selected menuitem in the list menu – using index & start variable in listmenu\_t. Will return 0x0 (NUL) if no item is currently selected.

### Returns

menuitem\_t\* retMenuItem : Pointer to a menuitem\_t structure

5.28.2.21 menurect\_t\* LCDListBox ( int pos )

Get the frame info of a specific list item in form of a menurect\_t structure.

### **Parameters**

```
int pos : Index of list item
```

### Returns

menurect\_t\* retMenuRect : Pointer to a menurect\_t structure

5.28.2.22 int LCDListCount (void)

Get the number of list items supported by the current display (text) configuration. This includes the item for title bar - thus, different from count variable in listmenu\_t as updated by an LCDList() call.

int listCount: Number of list items (including title box)

5.28.2.23 int LCDListIndex (int index)

Set the list index.

#### **Parameters**

int	index : List index
"""	mack: Elet mack

### Returns

int retVal: List index

5.28.2.24 int LCDListScrollDown (void)

Scrolls the list display down. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

### Returns

int retVal: always 0

5.28.2.25 int LCDListScrollUp (void )

Scrolls the list display up. Menu index is not altered. If the active menu item goes out of focus, the index becomes negative (no item selected).

#### Returns

int retVal : always 0

5.28.2.26 int LCDMenu ( char \* string1, char \* string2, char \* string3, char \* string4 )

Set menu entries in KEY\_CLASSIC mode (4-buttons). Also sets the LCD\_SHOWMENU flag and refresh the LCD.

char*	string1 : Menu entry for KEY1 in classic mode
char*	string2 : Menu entry for KEY2 in classic mode
char*	string3 : Menu entry for KEY3 in classic mode
char*	string4 : Menu entry for KEY4 in classic mode

int retVal: always 0

5.28.2.27 int LCDMenul ( int pos, char \* string,  $rgb_t fgcol$ ,  $rgb_t bgcol$ , void \* userp )

Set specific menu entry in KEY\_CLASSIC mode (index given by pos). Color customization for specific key is now possible (fgcol/bgcol). A user-specific data can be linked to the menu using userp pointer. Will also set the LCD\_SHOWMENU flag and refresh the LCD.

#### **Parameters**

int	pos : Select menu entry in classic mode
char*	string: Menu entry for the key at specified index
rgb_t	fgcol: Textcolor for the menu
rgb_t	bgcol : Background color for the menu
void*	userp: A general purpose pointer for user-specific data

#### Returns

int retVal: always 0

5.28.2.28 menuitem\_t\* LCDMenuItem ( int index )

Return the menuitem at a given position.

### **Parameters**

- 1	:a	incluses and the company it and
- 1	ınt	Index : position of the menuitem
- 1		

### **Returns**

menuitem\_t\* menuItem

5.28.2.29 int LCDNeedRefresh (void)

Indicate if the LCD need to be refreshed.

#### Returns

int retVal: non-null value indicate that the LCD need to be refreshed

5.28.2.30 int LCDPrintf (const char \* format, ...)

Print formatted string to LCD and refresh LCD. Cursor position is updated.

### **Parameters**

const   char∗ format : Formatted string	const   char* format : Formattee	string
---	----------------------------------	--------

### Returns

int retVal: always 0

## 5.28.2.31 int LCDPutChar ( char c )

Write a character to LCD and refresh LCD. Cursor position is updated.

### **Parameters**

char	c : Character to be displayed
0	o i o i a a a a a a a a a a a a a a a a

### Returns

int retVal: always 0

## 5.28.2.32 int LCDPutFloat (float val)

Print floating-point value to LCD and refresh LCD. Cursor position is updated.

### **Parameters**

int   val : Floating-point value to be displayed	
IIIL Val . I Idaliiu-boiiil value lo be disblaved	

### Returns

int retVal: always 0

## 5.28.2.33 int LCDPutFloatS ( float val, int spaces, int decimals )

Print floating-point value to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

int	val : Floating-point value to be displayed
int	spaces : Text space for the integer
int	decimals : Number of decimal points to display

int retVal: always 0

5.28.2.34 int LCDPutHex (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

### **Parameters**

int	val : Hex number to be displayed
-----	----------------------------------

### **Returns**

int retVal: always 0

5.28.2.35 int LCDPutHex1 (int val)

Print hexadecimal number to LCD and refresh LCD. Cursor position is updated. Utilize LCDPrintf for conversion.

### **Parameters**

int	val : Hex number to be displayed
-----	----------------------------------

#### **Returns**

int retVal: always 0

5.28.2.36 int LCDPutImageRGB (int xpos, int ypos, int xsize, int ysize, byte\_t \* data )

Place a RGB color image (24bpp) at (xpos,ypos) position on the LCD screen.

### Parameters

	int	xpos : X-coordinate of top-left image position
Ī	int	ypos: Y-coordinate of top-left image position
Ī	int	xsize : Image width
	int	ysize : Image height
Ī	byte_t*	data : Pointer to image data (24-bit per pixel)

### Returns

int retVal: always 0

5.28.2.37 int LCDPutInt (int val)

Print integer to LCD and refresh LCD. Cursor position is updated.

### **Parameters**

```
int val : Integer to be displayed
```

#### Returns

int retVal: always 0

5.28.2.38 int LCDPutIntS (int val, int spaces)

Print integer to LCD and refresh LCD. Cursor position is updated. Text space usage can be specified (formatting).

### **Parameters**

int	val : Integer to be displayed
int	spaces : Text space for the integer

#### Returns

int retVal: always 0

5.28.2.39 int LCDPutString ( char \* string )

Print string to LCD and refresh LCD. Cursor position is updated.

### **Parameters**

char*	string : String to be displayed

### Returns

int retVal: always 0

5.28.2.40 int LCDRefresh (void)

Refresh the screen (i.e write display buffers to the framebuffer device).

### Returns

int retVal: always 0

5.28.2.41 int LCDRelease ( )

Release the LCD.

**Returns** 

int retVal: always 0

5.28.2.42 int LCDResetMode ( hword\_t mode )

Reset the internal mode flag bits to a previously saved mode.

### **Parameters**

-		
- 1	, , ,	L M L (L L'EVOD L
- 1	nword t	mode : Mode flag - bit XORed
- 1	TIVVOIG L	Initioue . Widde mag - bit AOI led

### **Returns**

int retVal: always 0

5.28.2.43 int LCDSetChar (int row, int column, char c)

LCDPutChar with text position specified.

### **Parameters**

int	row : Cursor position
int	column : Cursor position
char	c : Character to be displayed

#### **Returns**

int retVal: always 0

5.28.2.44 int LCDSetList ( listmenu\_t \* menulist )

Unlike LCDList(), this will blindly assign menulist to the mainlist for display. Doesn't update anything in the menulist structure, nor modify any internal flags. Useful to maintain multiple lists fo menu display.

listmenu t*	menulist : Listmenu to be used for display
nounona_t-	monand: Eletinona to be assa for alepiay

int retVal: always 0

## 5.28.2.45 int LCDSetMode ( hword\_t mode )

Update the internal mode flag bits.

### **Parameters**

hword t mode: LCD Mode flag	
hword t mode: I (:I) Mode flag	$\neg$
mora i mode. Lob mode mag	

### Returns

int retVal: always 0

## 5.28.2.46 int LCDSetPixel (int x, int y, rgb\_t color)

Sets the color of the pixel at (x,y) coordinate to color.

### **Parameters**

int	x : X-coordinate of the pixel
int	y: Y-coordinate of the pixel
rgb_t	color : RGB color value for the pixel

## Returns

int retVal: always 0

## 5.28.2.47 int LCDSetPos (int row, int column)

Set the text cursor position to (row, column).

int	row : Text cursor row index
int	column : Text cursor column index

int retVal: always 0

5.28.2.48 int LCDSetPrintf ( int row, int column, const char \* format, ... )

LCDPrintf with text position specified.

#### **Parameters**

int	row : Cursor position
int	column : Cursor position
const	char* format : Formatted string

#### **Returns**

int retVal: always 0

5.28.2.49 int LCDSetString (int row, int column, char \* string)

LCDPutString with text position specified.

### **Parameters**

int	row : Cursor position
int	column : Cursor position
char*	c : String to be displayed

### Returns

int retVal: always 0

5.28.2.50 rect\_t LCDTextBar ( int row, int column, int length, int fill, rgb\_t color )

Draw a textbar for text starting at position (row, column) until (row, column+length). - The textbar will take about 25%-50% of text height & width to draw its frame. The fill parameter will define how much of the text bar should be 'filled' with color (like a progress bar).

int	row : Start text cursor position
int	column : Start text cursor position
int	length: Text length of the bar
int	fill : Percentage of textbar to be filled
rgb_t	color : Fill color for the textbar

rect\_t rect : rect\_t structure for the textbar's frame

5.28.2.51 int LCDTextColor ( rgb\_t fgcol, rgb\_t bgcol, char colorflags )

Set the default color for text (including background) and related flags (e.g. for transparent background).

### **Parameters**

rgb_t	fgcol : Default color for text
rgb_t	bgcol : Default color for text background
char	colorflags : Mode flag for text color

## Valid value for colorflags:

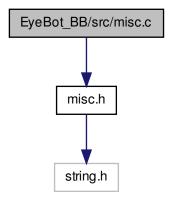
- LCD\_BGCOL\_TRANSPARENT
- LCD\_BGCOL\_INVERSE
- LCD\_FGCOL\_INVERSE
- LCD\_BGCOL\_NOTRANSPARE
- LCD\_BGCOL\_NOINVERSE
- LCD\_FGCOL\_NOINVERSE

int retVal: always 0

# 5.29 EyeBot\_BB/src/misc.c File Reference

Defines misc functions.

#include "misc.h" Include dependency graph for misc.c:



## **Functions**

• void **strcpy\_n** (char \*\_\_dest, const char \*\_\_src, size\_t \_\_n)

## 5.29.1 Detailed Description

Defines misc functions.

**Author** 

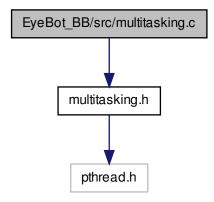
Remi KEAT

# 5.30 EyeBot\_BB/src/multitasking.c File Reference

Defines multitasking functions.

154 File Documentation

#include "multitasking.h" Include dependency graph for multitasking.c:



## 5.30.1 Detailed Description

Defines multitasking functions.

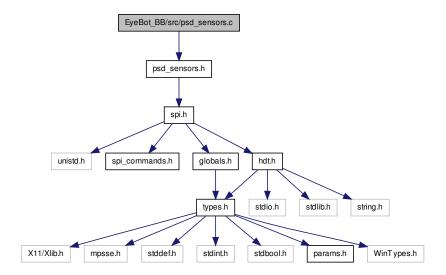
Author

Remi KEAT

# 5.31 EyeBot\_BB/src/psd\_sensors.c File Reference

Defines functions to use the PSD sensors.

#include "psd\_sensors.h" Include dependency graph for psd\_sensors.c:



### **Functions**

- PSDHandle PSDInit (DeviceSemantics semantics)

  Initialize single PSD with given semantics. Up to 8 PSDs can be initialized.
- int PSDGetRaw (PSDHandle psd)

Delivers raw-data measured by the selected PSD.

• int PSDGet (PSDHandle psd)

Delivers actual timestamp or distance measured by the selected PSD. If the raw reading is out of range for the given sensor, PSD\_OUT\_OF\_RANGE (=9999) is returned.

• int PSDRelease (PSDHandle psd)

Stops measurings and releases a PSD.

## 5.31.1 Detailed Description

Defines functions to use the PSD sensors.

Author

Remi KEAT

## 5.31.2 Function Documentation

### 5.31.2.1 int PSDGet ( PSDHandle psd )

Delivers actual timestamp or distance measured by the selected PSD. If the raw reading is out of range for the given sensor, PSD\_OUT\_OF\_RANGE (=9999) is returned.

#### **Parameters**

156

PSDHandle psd: the number of the psd to read	
--	--

#### Returns

int retVal: actual distance in mm (converted through internal table)

### 5.31.2.2 int PSDGetRaw ( PSDHandle psd )

Delivers raw-data measured by the selected PSD.

#### **Parameters**

PSDHandle	psd : Handle of the psd to read

### Returns

int readVal: actual raw-data (not converted)

## 5.31.2.3 PSDHandle PSDInit ( DeviceSemantics semantics )

Initialize single PSD with given semantics. Up to 8 PSDs can be initialized.

#### **Parameters**

Device-	semantics : unique definition for desired PSD
Semantics	

### Returns

PSDHandle psdHandle: unique handle for all further operations

## 5.31.2.4 int PSDRelease ( PSDHandle psd )

Stops measurings and releases a PSD.

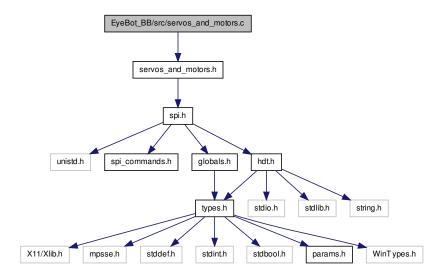
PSDHandle	psd
-----------	-----

int retVal: always 0

# 5.32 EyeBot\_BB/src/servos\_and\_motors.c File Reference

Defines functions to control servos and motors.

#include "servos\_and\_motors.h" Include dependency graph for servos\_ and\_motors.c:



### **Functions**

• SERVOHandle SERVOInit (DeviceSemantics semantics)

Initialize given servo.

• int SERVORelease (SERVOHandle handle)

Release given servos.

• int SERVOSet (SERVOHandle handle, int angle)

Set the given servos to the same given angle.

• MOTORHandle MOTORInit (DeviceSemantics semantics)

Initialize given motor.

• int MOTORRelease (MOTORHandle handle)

Release given motor.

int MOTORDrive (MOTORHandle handle, int speed)

Set the given motors to the same given speed.

• QUADHandle QUADInit (DeviceSemantics semantics)

Initialize given Quadrature-Decoder (up to 8 decoders are possible)

long QUADRead (QUADHandle handle)

Read actual Quadrature-Decoder counter, initially zero.

int QUADReset (QUADHandle handle)

Reset one or more Quadrature-Decoder.

• int QUADRelease (QUADHandle handle)

Release one or more Quadrature-Decoder.

DeviceSemantics QUADGetMotor (QUADHandle handle)

Get the semantic of the corresponding motor.

## 5.32.1 Detailed Description

Defines functions to control servos and motors.

**Author** 

Remi KEAT

### 5.32.2 Function Documentation

5.32.2.1 int MOTORDrive ( MOTORHandle handle, int speed )

Set the given motors to the same given speed.

#### **Parameters**

MOTOR-	handle
Handle	
int	speed : motor speed in percent

Valid values for speed:

- · -100 to 100 (full backward to full forward)
- 0 for full stop

### Returns

int retVal: always 0

### 5.32.2.2 MOTORHandle MOTORInit ( DeviceSemantics semantics )

Initialize given motor.

### **Parameters**

Device-	semantics
Semantics	

### **Returns**

MOTORHandle motorHandle

## 5.32.2.3 int MOTORRelease ( MOTORHandle handle )

Release given motor.

### **Parameters**

MOTOR-	handle
Handle	

### Returns

int retVal: always 0

## 5.32.2.4 DeviceSemantics QUADGetMotor ( QUADHandle handle )

Get the semantic of the corresponding motor.

### **Parameters**

QUAD-	handle : ONE decoder-handle
Handle	

0 = wrong handle

### Returns

DeviceSemantics semantic: Of the corresponding motor

## 5.32.2.5 QUADHandle QUADInit ( DeviceSemantics semantics )

Initialize given Quadrature-Decoder (up to 8 decoders are possible)

Device-	semantics
Semantics	

QUADHandle handle: QUADHandle or 0 for error

## 5.32.2.6 long QUADRead ( QUADHandle handle )

Read actual Quadrature-Decoder counter, initially zero.

#### **Parameters**

QUAD-	handle : ONE decoder-handle
Handle	

### Returns

long value of the encoder

## 5.32.2.7 int QUADRelease ( QUADHandle handle )

Release one or more Quadrature-Decoder.

### **Parameters**

QUAD-	handle: logical-or of decoder-handles to be released
Handle	

## Returns

```
int retVal : 0 = ok
-1 = error wrong handle
```

## 5.32.2.8 int QUADReset ( QUADHandle handle )

Reset one or more Quadrature-Decoder.

## **Parameters**

QUAD-	handle: logical-or of decoder-handles to be reseted
Handle	

### Returns

```
int retVal : 0 = ok
-1 = error wrong handle
```

## 5.32.2.9 SERVOHandle SERVOInit ( DeviceSemantics semantics )

Initialize given servo.

### **Parameters**

Device-	semantics
Semantics	

#### Returns

SERVOHandle servoHandle

## 5.32.2.10 int SERVORelease ( SERVOHandle handle )

Release given servos.

### **Parameters**

SERVO-	handle
Handle	

### **Returns**

int retVal: always 0

## 5.32.2.11 int SERVOSet ( SERVOHandle handle, int angle )

Set the given servos to the same given angle.

### **Parameters**

SERVO-	handle
Handle	
int	angle : valid values = 0-360

### Returns

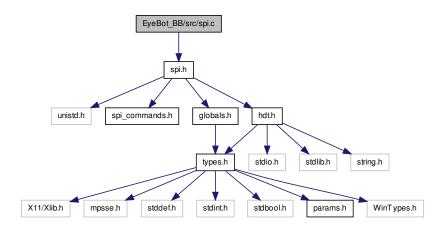
int retVal: always 0

# 5.33 EyeBot\_BB/src/spi.c File Reference

Defines fonctions for sending and receiving SPI messages.

162 File Documentation

#include "spi.h" Include dependency graph for spi.c:



### **Functions**

• int checkError (SPIHandle handle)

Check if an error happened if so print error message.

• SPIHandle SPIInit (int deviceNumber)

Initialize the SPI device.

• int SPIRelease (SPIHandle spiHandle)

Release the SPI device.

- int SPISend (SPIHandle spiHandle, size\_t length, const uint8\_t data[])
- Send a SPI message.
   int SPIRead (SPIHandle spiHandle, size\_t length, uint8\_t \*data[])

Read a SPI message.

• int SPIReadDefault (size t length, uint8 t \*data[])

Read a SPI message on the default SPI device.

• int SPISendDefault (size\_t length, const uint8\_t data[])

Send a SPI message on the default SPI device.

### 5.33.1 Detailed Description

Defines fonctions for sending and receiving SPI messages.

### Author

Remi KEAT

### 5.33.2 Function Documentation

5.33.2.1 int checkError ( SPIHandle handle )

Check if an error happened if so print error message.

### **Parameters**

SPIHandle	handle

### Returns

int retVal: non-null value if an error happened

5.33.2.2 SPIHandle SPIInit (int deviceNumber)

Initialize the SPI device.

### **Parameters**

int	deviceNumber
-----	--------------

### Returns

SPIHandle spiHandle

5.33.2.3 int SPIRead ( SPIHandle spiHandle, size\_t length, uint8\_t \* data[] )

Read a SPI message.

#### **Parameters**

SPIHandle	spiHandle
size_t	length
uint8_t*	data[]

#### **Returns**

int retVal: always 0

5.33.2.4 int SPIReadDefault ( size\_t length, uint8\_t \* data[] )

Read a SPI message on the default SPI device.

### **Parameters**

size_t	length
uint8_t*	data[]

### Returns

int retVal: always 0

5.33.2.5 int SPIRelease (SPIHandle spiHandle)

Release the SPI device.

### **Parameters**

SPIHandle	spiHandle

### Returns

int retVal: always 0

5.33.2.6 int SPISend ( SPIHandle spiHandle, size\_t length, const uint8\_t data[] )

Send a SPI message.

### **Parameters**

SPIHandle	spiHandle
size_t	length
const	uint8_t data[]

### Returns

int retVal: always 0

5.33.2.7 int SPISendDefault ( size\_t length, const uint8\_t data[] )

Send a SPI message on the default SPI device.

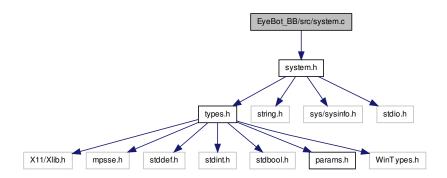
size_t	length
const	uint8_t data[]

int retVal: always 0

## 5.34 EyeBot\_BB/src/system.c File Reference

Defines functions for the system.

#include "system.h" Include dependency graph for system.c:



### **Functions**

- char \* execute (char \*command)
- char \* OSVersion (void)

Returns string containing running RoBIOS version.

• int OSMachineSpeed (void)

Inform the user how fast the processor runs.

int OSMachineType (void)

Inform the user in which environment the program runs.

char \* OSMachineName (void)

Inform the user with which name the Eyebot is titled.

unsigned char OSMachineID (void)

Inform the user with which ID the Eyebot is titled.

• int OSError (char \*msg, int number, bool deadend)

Print message and number to display then stop processor (deadend) or wait for key.

• int OSInfoCPU (info\_cpu\_t \*infoCPU)

Collects infos about the CPU - name, speed, architecture and bogusMips.

int OSInfoMem (info\_mem\_t \*infoMem)

Collects infos about the memory.

int OSInfoProc (info\_proc\_t \*infoProc)

Collects infos about processes.

• int OSInfoMisc (info\_misc\_t \*infoMisc)

Collects system's miscellaneous infos - uptime, vbatt.

### 5.34.1 Detailed Description

Defines functions for the system.

**Author** 

Remi KEAT

## 5.34.2 Function Documentation

5.34.2.1 int OSError ( char \* msg, int number, bool deadend )

Print message and number to display then stop processor (deadend) or wait for key.

### **Parameters**

char*	msg : pointer to message
int	number : int number
BOOL	deadend : switch to choose deadend or keywait

Valid values are:

- 0 = no deadend
- 1 = deadend

### Returns

int retVal: Always 0

5.34.2.2 int OSInfoCPU ( info\_cpu\_t \* infoCPU )

Collects infos about the CPU – name, speed, architecture and bogusMips.

### **Parameters**

info\_cpu\_t\* infoCPU : pointer to a structure (info\_cpu\_t) containing the cpu infos

#### Returns

int retVal: always 0

5.34.2.3 int OSInfoMem ( info\_mem\_t \* infoMem )

Collects infos about the memory.

## **Parameters**

info_mem_t*	infoMem: pointer to a structure (info_mem_t) which contains the mem-
	ory infos

#### **Returns**

int retVal: always 0

5.34.2.4 int OSInfoMisc ( info\_misc\_t \* infoMisc )

Collects system's miscellaneous infos – uptime, vbatt.

### **Parameters**

info_misc_t	infoMisc: pointer to a structure (info_misc_t) which contains the misc
	infos

### Returns

int retVal: always 0

5.34.2.5 int OSInfoProc (  $info\_proc\_t * infoProc$  )

Collects infos about processes.

### **Parameters**

info_proc_t	infoProc : pointer to a structure (info_proc_t) which contains the process
	infos

### Returns

int retVal: always 0

5.34.2.6 unsigned char OSMachineID (void)

Inform the user with which ID the Eyebot is titled.

### Returns

unsigned char ID: ID of actual Eyebot

5.34.2.7 char\* OSMachineName (void)

Inform the user with which name the Eyebot is titled.

Returns

char\* machineName : Name of actual Eyebot

5.34.2.8 int OSMachineSpeed (void)

Inform the user how fast the processor runs.

Returns

int speed: actual clockrate of CPU in Hz

5.34.2.9 int OSMachineType (void)

Inform the user in which environment the program runs.

Returns

int machineType : Type of used hardware

Valid values are: VEHICLE, PLATFORM, WALKER

5.34.2.10 char\* OSVersion (void)

Returns string containing running RoBIOS version.

Returns

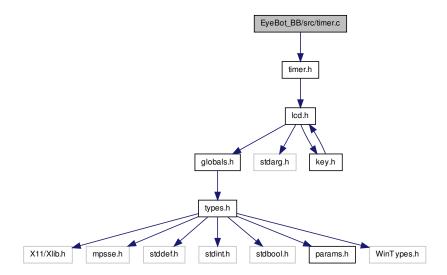
char\* version : OS version

Example: "3.1b"

# 5.35 EyeBot\_BB/src/timer.c File Reference

Defines functions related to the timer.

#include "timer.h" Include dependency graph for timer.c:



## **Functions**

• int OSWait (int n)

Busy loop for n\*1/100 seconds.

## 5.35.1 Detailed Description

Defines functions related to the timer.

Author

Remi KEAT

## 5.35.2 Function Documentation

5.35.2.1 int OSWait ( int *n* )

Busy loop for n\*1/100 seconds.

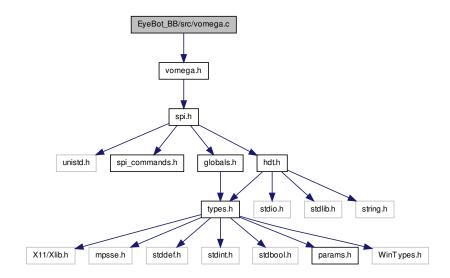
int	n : time to wait

int retVal: always 0

## 5.36 EyeBot\_BB/src/vomega.c File Reference

### Defines the VW functions.

#include "vomega.h" Include dependency graph for vomega.c:



## **Functions**

- VWHandle VWInit (DeviceSemantics semantics, int Timescale)

  Initialize given VW-Driver (only 1 can be initialized!). The motors and encoders are automatically reserved!! The Timescale allows to adjust the tradeoff between accuracy (scale=1, update at 100Hz) and speed(scale>1, update at 100/scale Hz).
- int VWDriveStraight ( $VWHandle\ handle\ ,$  meter delta, meter $PerSec\ v$ )

Drives distance "delta" with speed v straight ahead (forward or backward) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

- int VWDriveTurn (VWHandle handle, radians delta, radPerSec w)
  - Turns about "delta" with speed w on the spot (clockwise or counter-clockwise) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.
- int VWDriveWait (VWHandle handle)

Blocks the calling process until the previous VWDriveX() command has been completed.

### 5.36.1 Detailed Description

Defines the VW functions.

### Author

Remi KEAT

### 5.36.2 Function Documentation

## 5.36.2.1 int VWDriveStraight ( VWHandle handle, meter delta, meterPerSec v )

Drives distance "delta" with speed v straight ahead (forward or backward) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

#### **Parameters**

VWHandle	handle : ONE VWHandle
meter	delta : distance to drive in m
meterPer-	v : speed to drive with (always positive!)
Sec	

### delta:

- pos. -> forward
- $\bullet \ \ \text{neg.} \ -{>} \ \text{backward}$

#### Returns

int retVal:

0 = ok

-1 = error wrong handle

## 5.36.2.2 int VWDriveTurn ( VWHandle handle, radians delta, radPerSec w )

Turns about "delta" with speed w on the spot (clockwise or counter-clockwise) any subsequent call of VWDriveStraight, -Turn, -Curve or VWSetSpeed while this one is still being executed, results in an immediate interruption of this command.

#### **Parameters**

	VWHandle	handle : ONE VWHandle
ſ	radians	delta : degree to turn in radians
Γ	radPerSec	w : speed to turn with (always positive!)

### delta:

- pos. -> counter-clockwise
- · neg. -> clockwise

int retVal:

0 = ok

-1 = error wrong handle

### 5.36.2.3 int VWDriveWait ( VWHandle handle )

Blocks the calling process until the previous VWDriveX() command has been completed.

#### **Parameters**

VWHandle   handle : ONE VWHandle	
----------------------------------	--

### Returns

int retVal:

-1 = error wrong handle

0 = previous VWDriveX command has been completed

## 5.36.2.4 VWHandle VWInit ( DeviceSemantics semantics, int Timescale )

Initialize given VW-Driver (only 1 can be initialized!). The motors and encoders are automatically reserved!! The Timescale allows to adjust the tradeoff between accuracy (scale=1, update at 100Hz) and speed(scale>1, update at 100/scale Hz).

### **Parameters**

Device-	semantics
Semantics	
int	Timescale : prescale value for 100Hz IRQ (1 to)

### Returns

VWHandle handle: VWHandle or 0 for error