EyeBot_BB

Generated by Doxygen 1.7.6.1

Sat Sep 14 2013 08:41:43

Contents

Chapter 1

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.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 _HDTDevADC_t Struct Reference

Collaboration diagram for _HDTDevADC_t:

Public Attributes

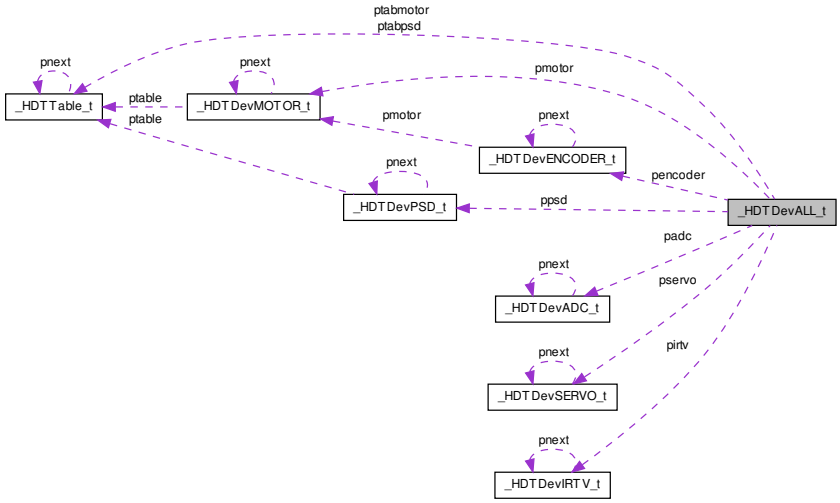
- struct _HDTDevADC_t * **pnext**
- char **name** [HDT_MAX_NAMECHAR]
- char **procname** [HDT_MAX_NAMECHAR]
- int **denom**

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

- EyeBot_BB/include/types.h

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**

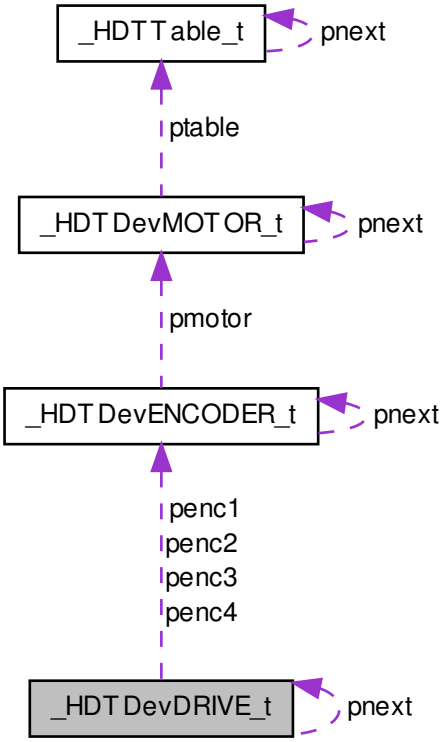
- char **name** [HDT_MAX_NAMECHAR]
- char **devname** [HDT_MAX_NAMECHAR]

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

- 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]

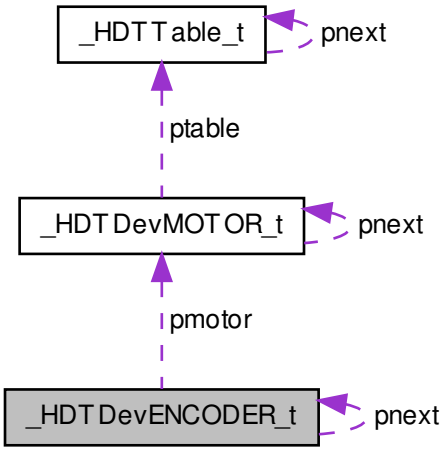
- char **encname1** [HDT_MAX_NAMECHAR]
- char **encname2** [HDT_MAX_NAMECHAR]
- char **encname3** [HDT_MAX_NAMECHAR]
- char **encname4** [HDT_MAX_NAMECHAR]
- HDTDevENCODER_t * **penc1**
- HDTDevENCODER_t * **penc2**
- HDTDevENCODER_t * **penc3**
- HDTDevENCODER_t * **penc4**
- int **drivetype**
- int **wheeldist1**
- int **axesdist**
- int **wheeldist2**

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

- EyeBot_BB/include/types.h

4.6 _HDTDevENCODER_t Struct Reference

Collaboration diagram for _HDTDevENCODER_t:



Public Attributes

- struct _HDTDevENCODER_t * **pnext**

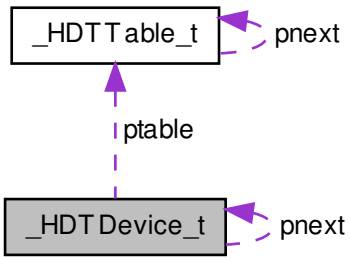
- char **name** [HDT_MAX_NAMECHAR]
- char **motorname** [HDT_MAX_NAMECHAR]
- HDTDevMOTOR_t * **pmotor**
- int **regaddr**
- int **clickspm**
- int **maxspeed**

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

- 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:

- EyeBot_BB/include/types.h

4.8 _HDTDevIRTV_t Struct Reference

Collaboration diagram for _HDTDevIRTV_t:

Public Attributes

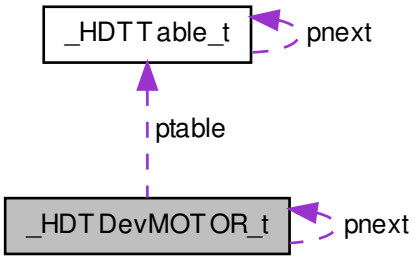
- struct _HDTDevIRTV_t * **pnext**
- char **name** [HDT_MAX_NAMECHAR]
- int **type**
- int **length**
- int **togmask**
- int **invmask**
- int **mode**
- int **buffsize**
- int **delay**

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

- EyeBot_BB/include/types.h

4.9 _HDTDevMOTOR_t Struct Reference

Collaboration diagram for _HDTDevMOTOR_t:



Public Attributes

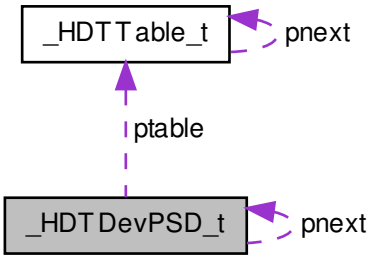
- struct _HDTDevMOTOR_t * **pnext**
- char **name** [HDT_MAX_NAMECHAR]
- char **tabname** [HDT_MAX_NAMECHAR]
- HDTTable_t * **ptable**
- int **regaddr**
- int **freq**

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

- EyeBot_BB/include/types.h

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

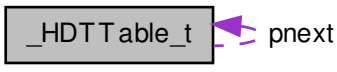
- int **length**
- char * **buffer**

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

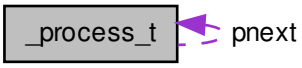
- char * **pTitle**

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

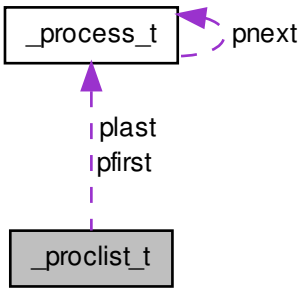
- pid_t **pid**
- char **name** [MAX_FILECHAR]
- struct _process_t * **pnext**

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**
- process_t * **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.

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

- EyeBot_BB/include/types.h

4.18 cursor_t Struct Reference

Public Attributes

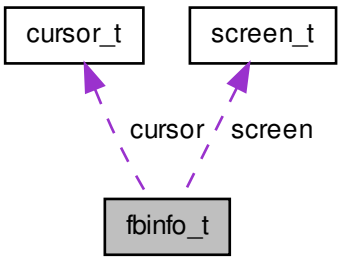
- int **x**
- int **y**
- int **xmax**
- int **ymax**

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:

- EyeBot_BB/include/types.h

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:

- EyeBot_BB/include/types.h

4.23 info_misc_t Struct Reference

Public 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:

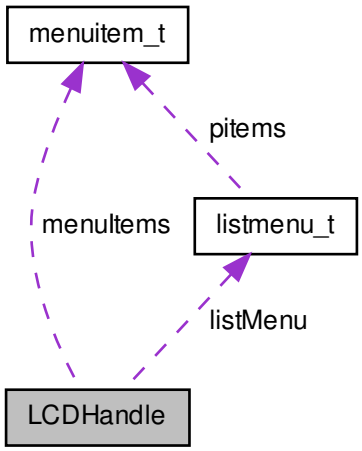
- EyeBot_BB/include/types.h

4.25 LCDHandle Struct Reference

Structure defining an LCD.

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

Collaboration diagram for LCDHandle:



Public Attributes

- int **lcdNum**
- 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 **menuItems** [4]
- listmenu_t * **listMenu**
- int **fd**
- bool **X11Error**

4.25.1 Detailed Description

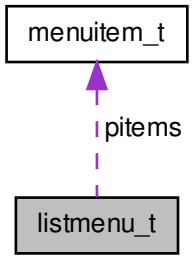
Structure defining an LCD.

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

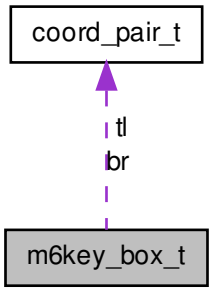
- char **title** [LCD_LIST_STRLENGTH]
- rgb_t **fgcol**
- rgb_t **bgcol**
- int **size**
- int **start**
- int **width**
- int **left**
- int **scroll**
- int **index**
- int **count**
- menuitem_t * **pitems**
- int **no_empty**

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

- EyeBot_BB/include/types.h

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:

- EyeBot_BB/include/types.h

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

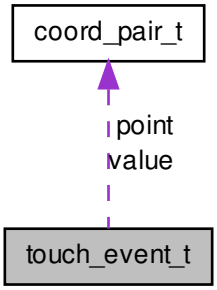
- int **xres**
- int **yres**
- int **bpp**

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

- EyeBot_BB/include/types.h

4.31 touch_event_t Struct Reference

Collaboration diagram for touch_event_t:



Public Attributes

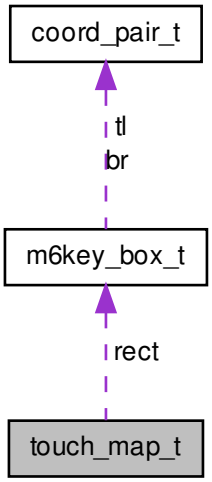
- coord_pair_t **point**
- coord_pair_t **value**
- int **sync**
- int **status**

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

- EyeBot_BB/include/types.h

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:

- EyeBot_BB/include/types.h

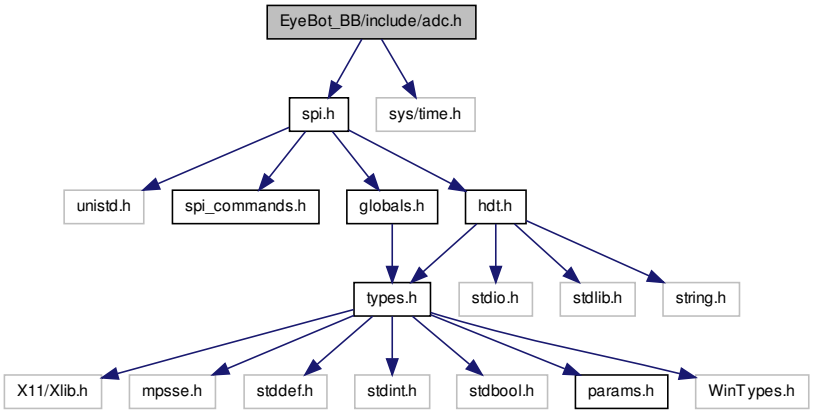
Chapter 5

File Documentation

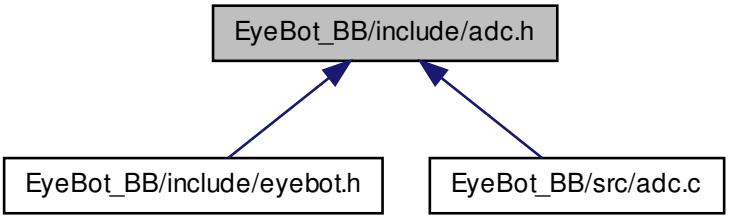
5.1 EyeBot_BB/include/adc.h File Reference

Header file for the ADC functions.

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



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 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

5.1.2.2 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	semantics : desired ADC channel

Returns

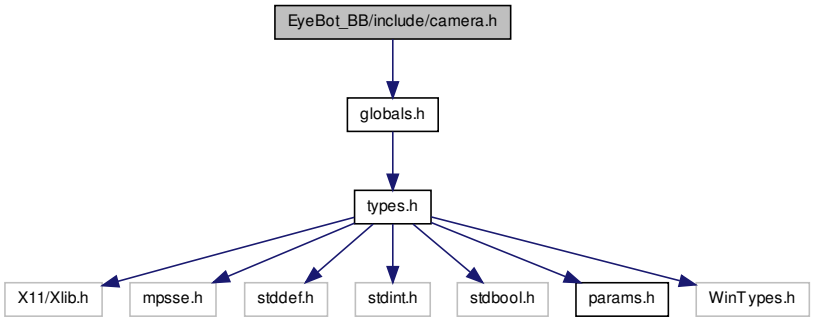
ADCHandle handle >0: Handler for the adc channel
0: Initialization error

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

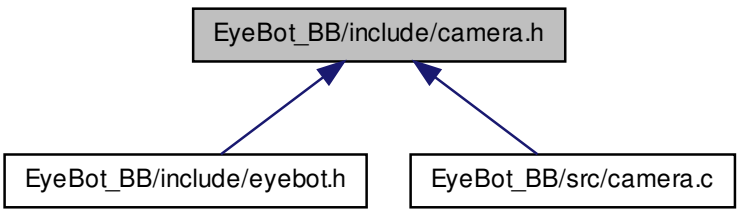
5.2 EyeBot_BB/include/camera.h File Reference

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
*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	semantics : handle of the desired camera

Returns

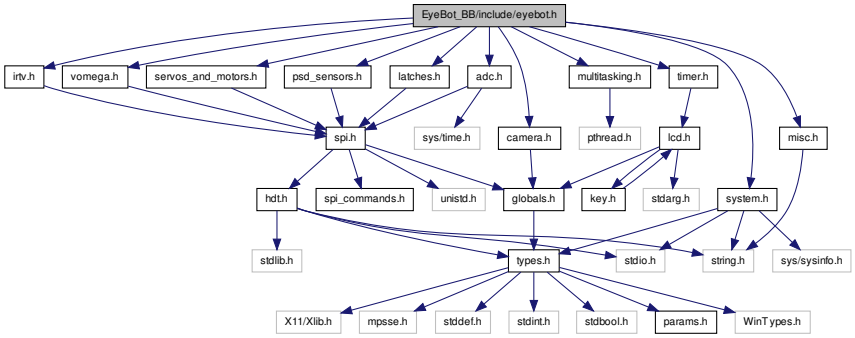
CAMHandle handle

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.

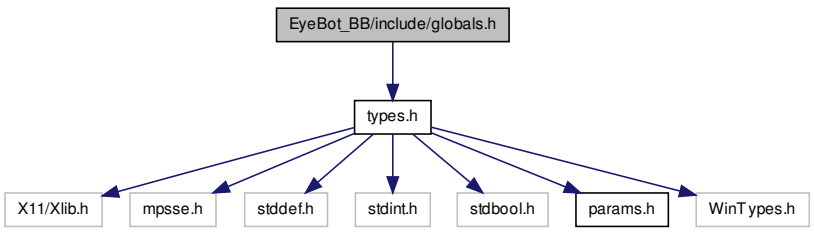
Author

Remi KEAT

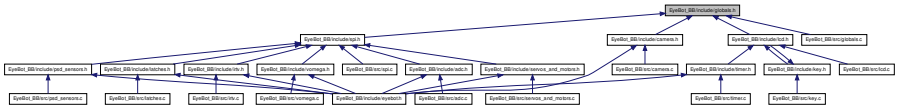
5.4 EyeBot_BB/include/globals.h File Reference

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.

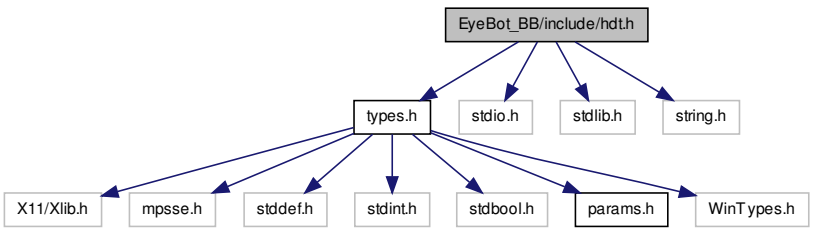
Author

Remi KEAT

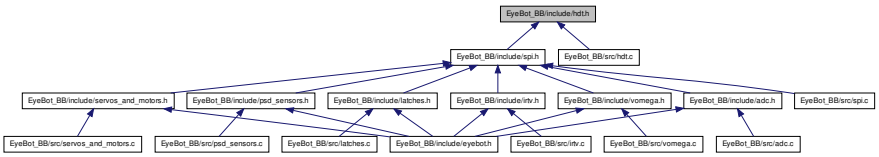
5.5 EyeBot_BB/include/hdt.h File Reference

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 <devices> 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 <devices> 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 <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 <devices> 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

*HDTDevADC_t**	pdevs : <device> list to be cleared

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

*HDTDevCAM_t**	pdevs : <device> list to be cleared

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

*HDTDevCOM_t**	pdevs : <device> list to be cleared

Returns

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

*HDTDevDRIVE_t**	pdevs : <device> list to be cleared

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

*HDTDevENCODER_t**	pdevs : <device> list to be cleared

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

*HDTDevIRTV_t**	pdevs : <device> list to be cleared

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

HDTDevMOTOR_t∗	pdevs : <device> list to be cleared

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

HDTDevPSD_t∗	pdevs : <device> list to be cleared

Returns

int retVal : always 0

5.5.2.9 int HDTClearSERVO (HDTDevSERVO_t ∗ *pdevs*)

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

Parameters

HDTDevSERVO_t∗	pdevs : <device> list to be cleared

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_t∗	ptables : tables to be cleared

Returns

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_t**	deventry : storage structure for the entry

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_t**	tabentry : storage structure for the table

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

HDTDevDRIVE_t∗	pdrives : list of drive methods
HDTDevENCODER_t∗	pencoders : list of encoders

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

HDTDevENCODER_t∗	pencoders : list of encoders
HDTDevMOTOR_t∗	pmotors : list of motors

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_t∗	deventry : storage structure for the entry
int	count : number of deventry storage supplied

Returns

int retVal :
-1 on failure
(number of entries) on success

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 <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

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

5.5.2.17 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.

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 <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

5.5.2.19 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 <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

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 <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

5.5.2.21 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 <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

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 <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

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

*char**	filename : hdt file to open
*HDTDevice_t**	pdevices : devices with tablename in linked list

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

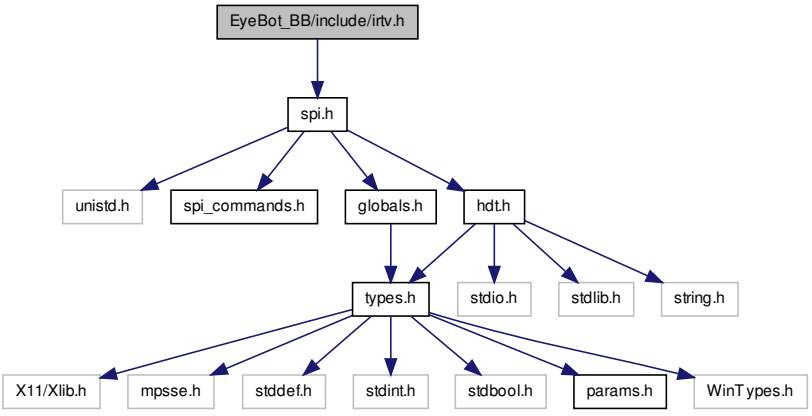
Returns

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

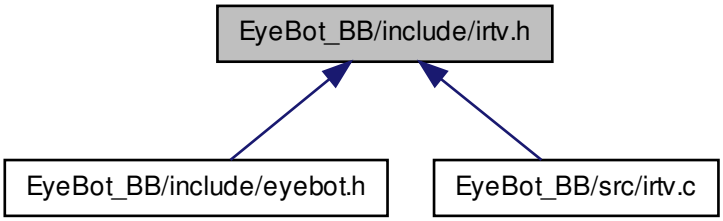
5.6 EyeBot_BB/include/irtv.h File Reference

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

Returns

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

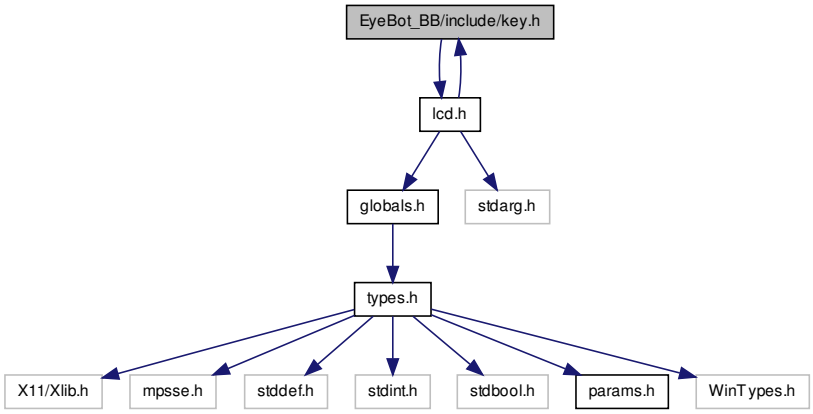
int retVal : Next code from the buffer

0 = no key

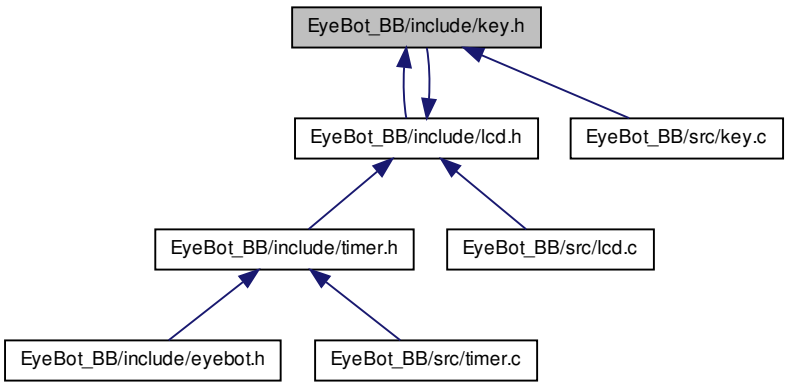
5.7 EyeBot_BB/include/key.h File Reference

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** 0xFFFFFFFF

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_REGIONS 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_t**	rawtouch : pointer to touch_event_t structure

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_t**	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_t**	region : Pointer to a storage for region data

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_t***	ptouch_map : Pointer to a touch_map_t structure

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

Returns

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

*touch_event_t**	rawtouch : pointer to touch_event_t structure this is optional! only if raw 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.

Returns

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_t**	region : Pointer to a region data

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

keymode_t	mode : Requested touch 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

keycode_t	excode : Expected keycode values (bit XORed)

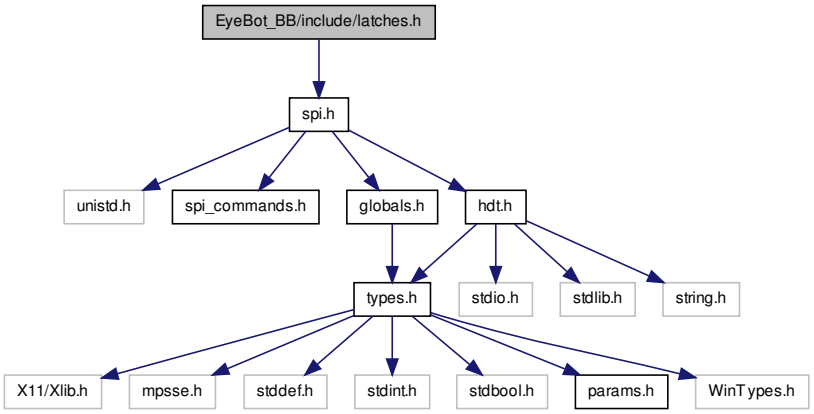
Returns

keycode_t retKey : Keycode value

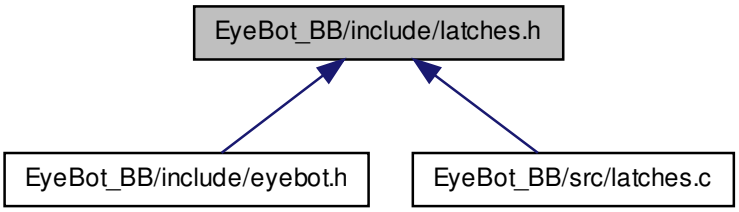
5.8 EyeBot_BB/include/latches.h File Reference

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

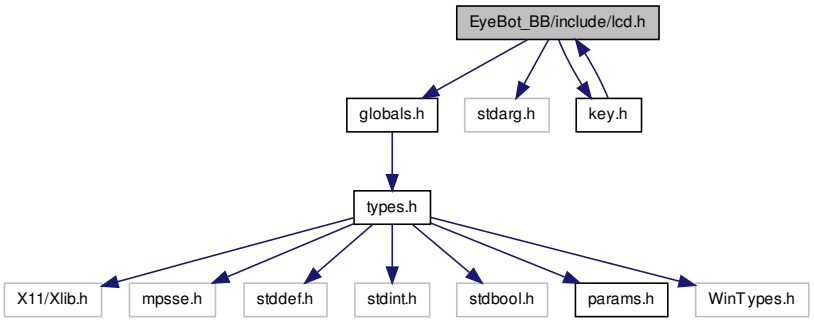
Returns

int retVal : always 0

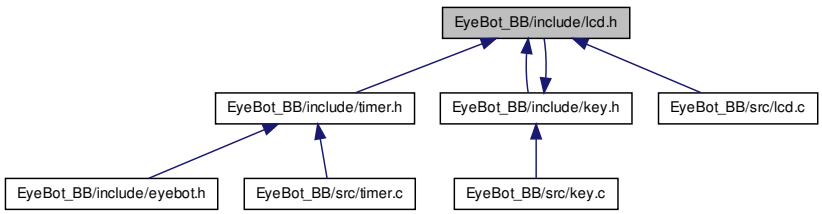
5.9 EyeBot_BB/include/lcd.h File Reference

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")

- #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("aquamarine")
- #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_SHOWMENU flag and refresh the LCD.

- int LCDMenuI (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).

- 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 LCDInit() 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.

Parameters

*char**	colorName

Returns

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.

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

5.9.2.5 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 LCDInit() for textsize.

Parameters

*fbinfo_t**	pinfo : Pointer to storage for screen & cursor info

Returns

int retVal
0 on success
Negative value on failure

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.

Returns

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

int	x : X-coordinate of the pixel
int	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.

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

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 listmenu_t 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.

Returns

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

5.9.2.20 int LCDListIndex (int *index*)

Set the list index.

Parameters

int	index : List index

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.

Parameters

*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

Returns

int retVal : always 0

5.9.2.24 int LCDMenuI (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

const	char* format : Formatted 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

char	c : Character 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

int	val : Floating-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).

Parameters

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

Returns

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	val : Hex number to be displayed

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

hword_t	mode : Mode flag - bit XORed

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.

Parameters

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

Returns

int retVal : always 0

5.9.2.42 int LCDSetMode (hword_t *mode*)

Update the internal mode flag bits.

Parameters

hword_t	mode : LCD Mode flag

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).

Parameters

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

Returns

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).

Parameters

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

Returns

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

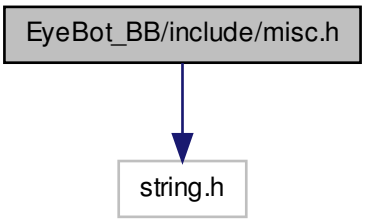
Returns

int retVal : always 0

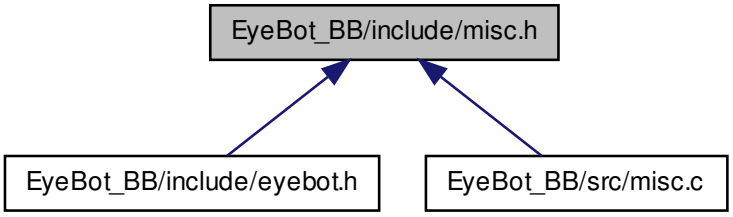
5.10 EyeBot_BB/include/misc.h File Reference

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.

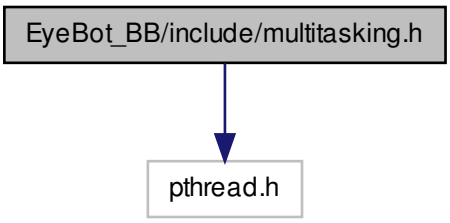
Author

Remi KEAT

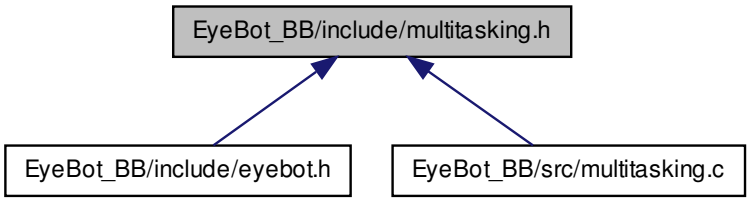
5.11 EyeBot_BB/include/multitasking.h File Reference

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.

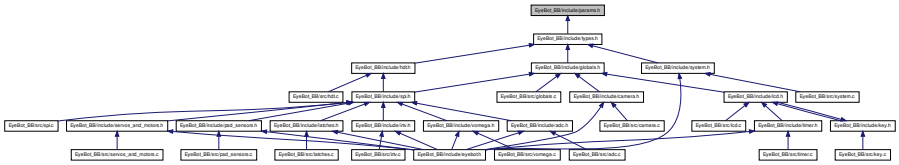
Author

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.

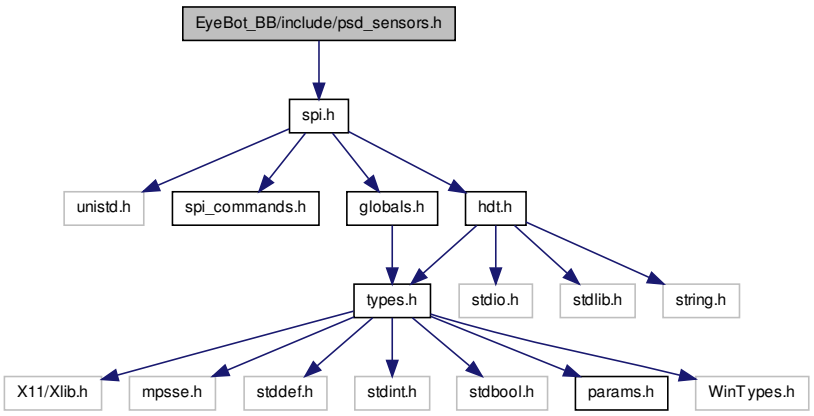
Author

Remi KEAT

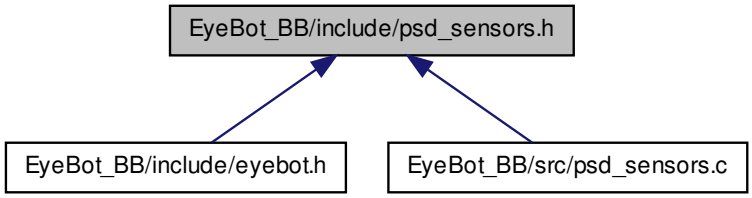
5.13 EyeBot_BB/include/psd_sensors.h File Reference

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	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.

Parameters

PSDHandle	psd : Handle of the psd to read

Returns

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	semantics : unique definition for desired PSD

Returns

PSDHandle psdHandle : unique handle for all further operations

5.13.2.4 int PSDRelease (PSDHandle *psd*)

Stops measurings and releases a PSD.

Parameters

PSDHandle	psd

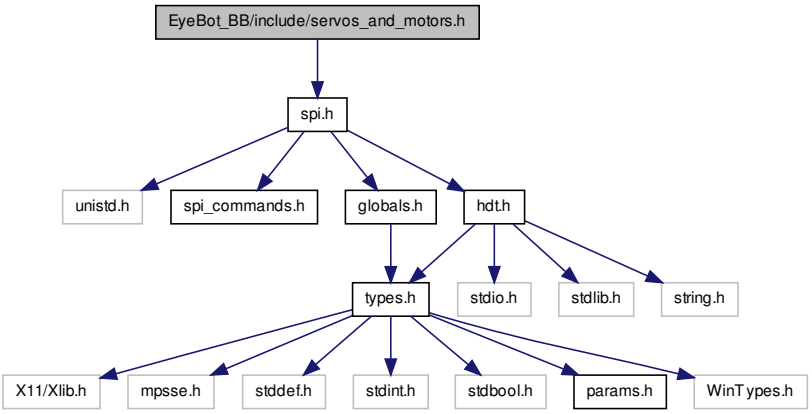
Returns

int retVal : always 0

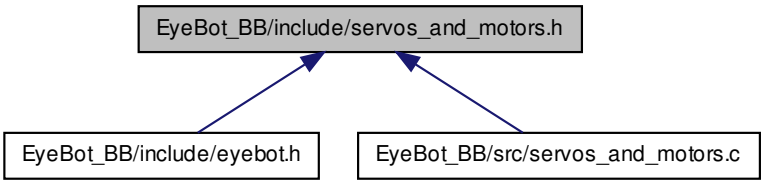
5.14 EyeBot_BB/include/servos_and_motors.h File Reference

Header file for the servos and motors functions.

`#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	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

5.14.2.2 MOTORHandle MOTORInit (DeviceSemantics *semantics*)

Initialize given motor.

Parameters

Device-Semantics	semantics

Returns

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	handle : ONE decoder-handle

Returns

long value of the encoder

5.14.2.5 int QUADRelease (QUADHandle *handle*)

Release one or more Quadrature-Decoder.

Parameters

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

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

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

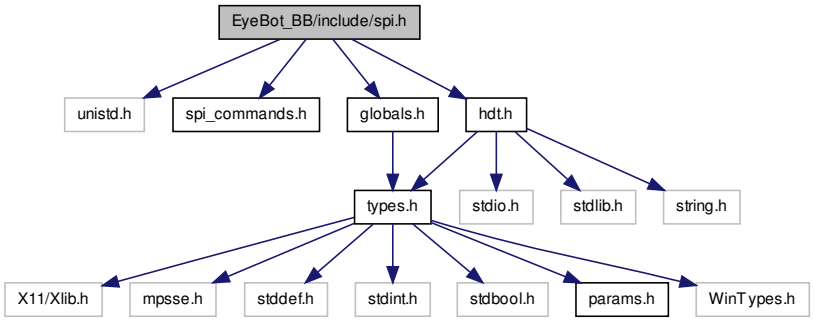
Returns

int retVal : always 0

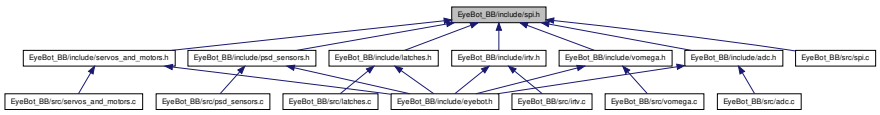
5.15 EyeBot_BB/include/spi.h File Reference

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 *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

5.15.2.3 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.

Parameters

size_t	length
const	uint8_t data[]

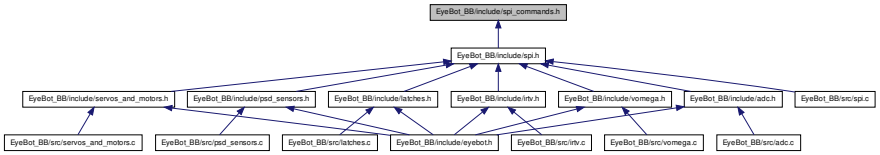
Returns

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.

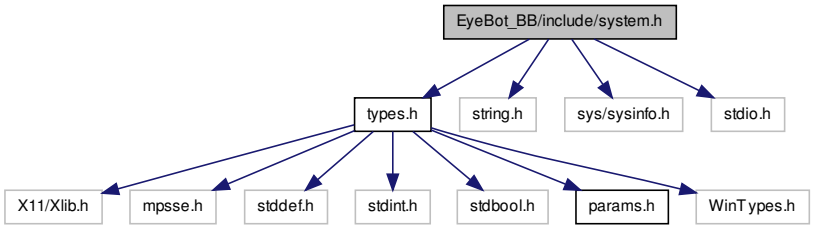
Author

Remi KEAT

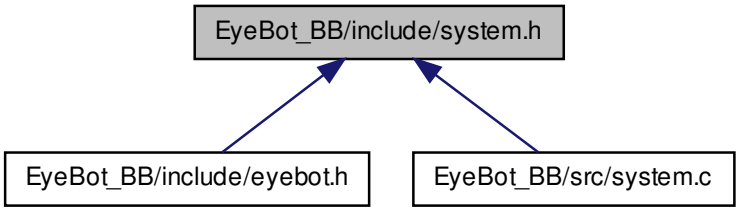
5.17 EyeBot_BB/include/system.h File Reference

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

*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

5.17.2.2 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 memory 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

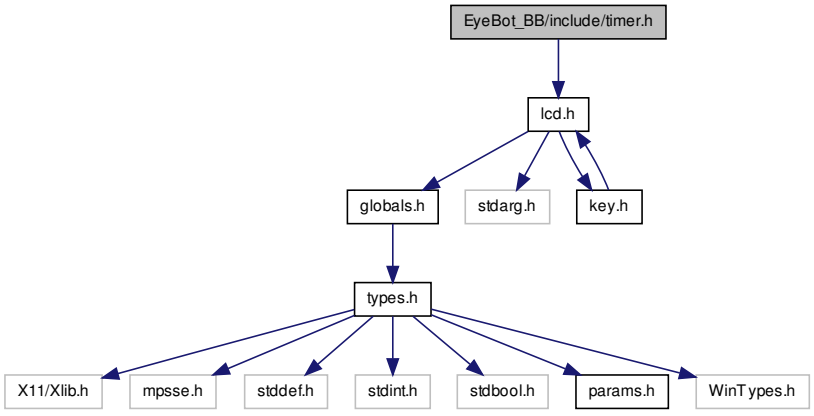
char* version : OS version

Example: "3.1b"

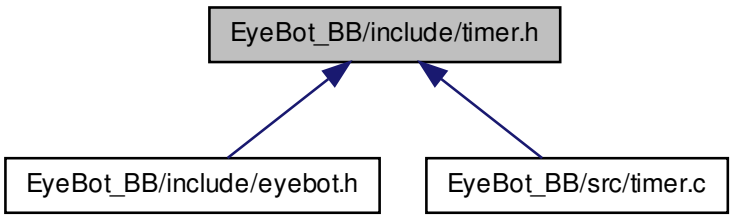
5.18 EyeBot_BB/include/timer.h File Reference

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

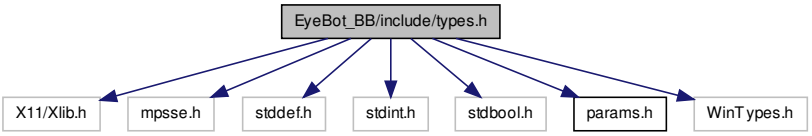
Returns

int retVal : always 0

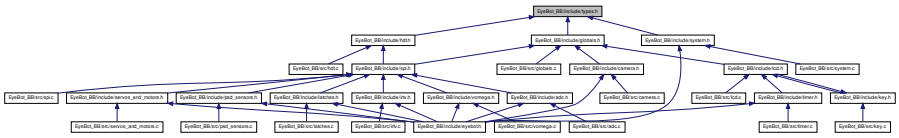
5.19 EyeBot_BB/include/types.h File Reference

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 **lcdmode_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 _HDTDevCAM_t **HDTDevCAM_t**
- typedef struct _HDTDevMOTOR_t **HDTDevMOTOR_t**
- typedef struct _HDTDevENCODER_t **HDTDevENCODER_t**
- typedef struct _HDTDevSERVO_t **HDTDevSERVO_t**
- typedef struct _HDTDevPSD_t **HDTDevPSD_t**
- typedef struct _HDTDevDRIVE_t **HDTDevDRIVE_t**
- typedef struct _HDTDevIRTV_t **HDTDevIRTV_t**
- typedef struct _HDTDevADC_t **HDTDevADC_t**
- typedef struct _HDTDevCOM_t **HDTDevCOM_t**

5.19.1 Detailed Description

Defines types.

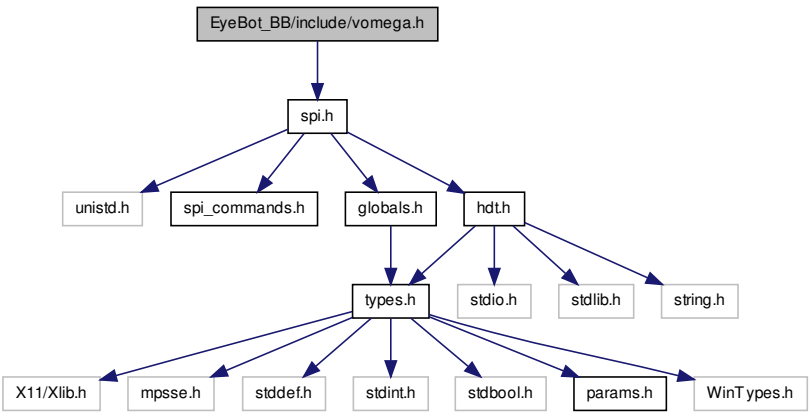
Author

Remi KEAT

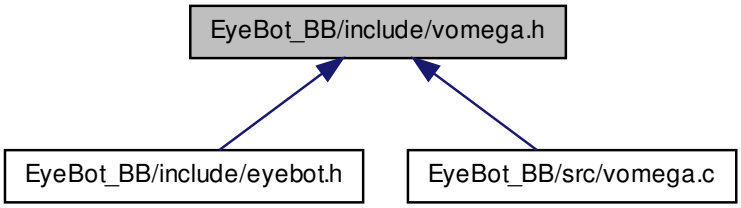
5.20 EyeBot_BB/include/vomega.h File Reference

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

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
meterPerSec	v : speed to drive with (always positive!)

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 ...)

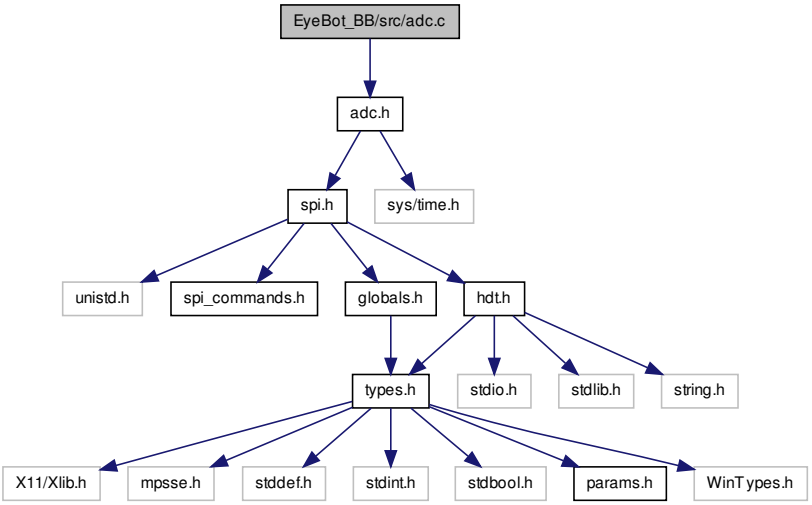
Returns

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	semantics : desired ADC channel

Returns

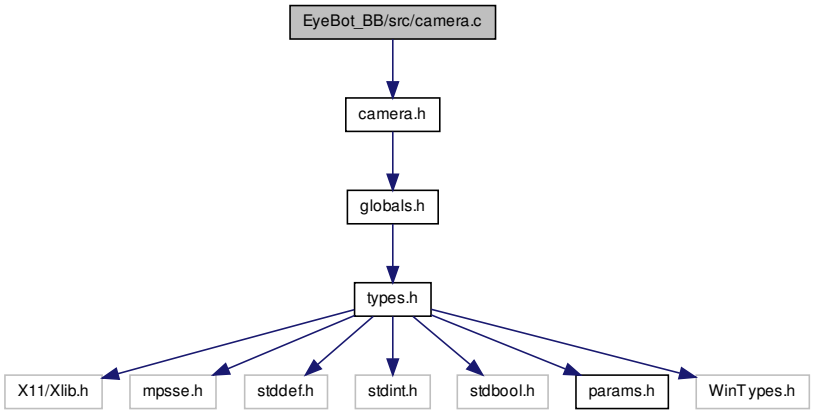
ADCHandle handle >0: Handler for the adc channel
0: Initialization error

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	semantics : handle of the desired camera

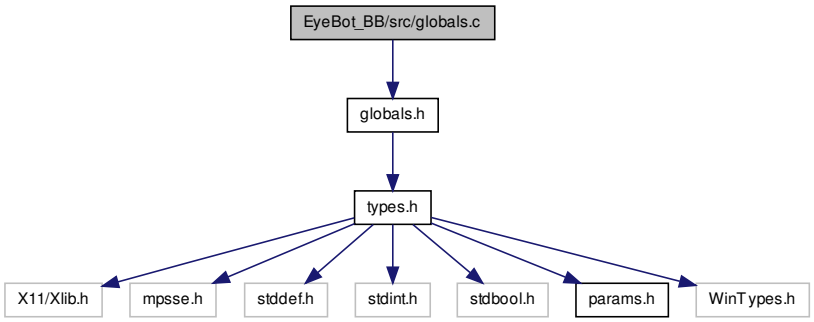
Returns

CAMHandle handle

5.23 EyeBot_BB/src/globals.c File Reference

Defines global variables.

`#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.

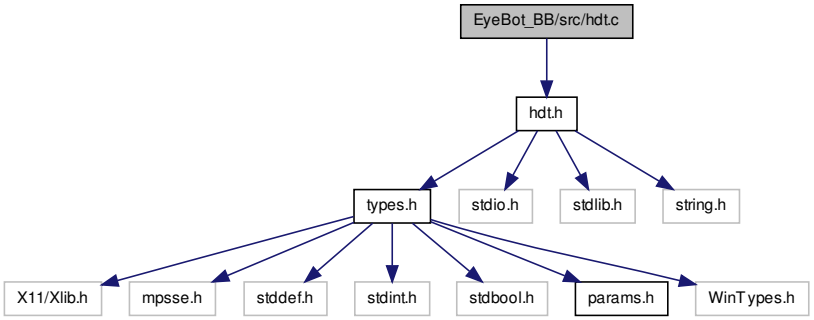
Author

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 HDTClearTable 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 <devices> 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 <devices> 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 <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 <devices> 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

*HDTDevADC_t**	pdevs : <device> list to be cleared

Returns

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

*HDTDevCAM_t**	pdevs : <device> list to be cleared

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

*HDTDevCOM_t**	pdevs : <device> list to be cleared

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

*HDTDevDRIVE_t**	pdevs : <device> list to be cleared

Returns

int retVal : always 0

5.24.2.6 int HDTClearENCODER (HDTDevENCODER_t * *pdevs*)

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

Parameters

HDTDevENCODER_t∗	pdevs : <device> list to be cleared

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

HDTDevIRTV_t∗	pdevs : <device> list to be cleared

Returns

int retVal : always 0

5.24.2.8 int HDTClearMOTOR (HDTDevMOTOR_t ∗ *pdevs*)

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

Parameters

HDTDevMOTOR_t∗	pdevs : <device> list to be cleared

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

HDTDevPSD_t∗	pdevs : <device> list to be cleared

Returns

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

*HDTDevSERVO_t**	pdevs : <device> list to be cleared

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_t**	ptables : tables to be cleared

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_t**	deventry : storage structure for the entry

Returns

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_t**	tabentry : storage structure for the table

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

*HDTDevDRIVE_t**	pdrives : list of drive methods
*HDTDevENCODER_t**	pencoders : list of encoders

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

HDTDevENCODER_t∗	pencoders : list of encoders
HDTDevMOTOR_t∗	pmotors : list of motors

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_t∗	deventry : storage structure for the entry
int	count : number of deventry storage supplied

Returns

int retVal :
-1 on failure
(number of entries) on success

5.24.2.17 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.

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 <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

5.24.2.19 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 <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 <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

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 <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

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 <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

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 <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

5.24.2.26 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

*char**	filename : hdt file to open
*HDTDevice_t**	pdevices : devices with tablename in linked list

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

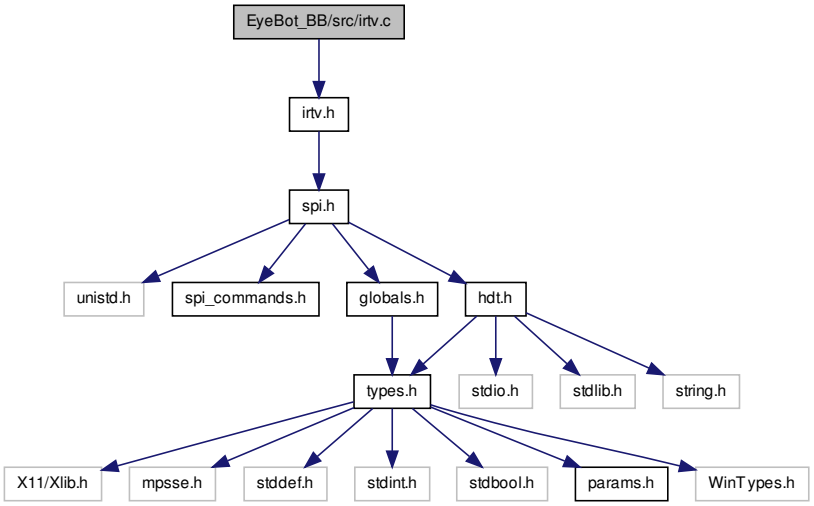
Returns

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 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.

5.25.1 Detailed Description

Defines IRTV functions.

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

Device-Semantics	semantics

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

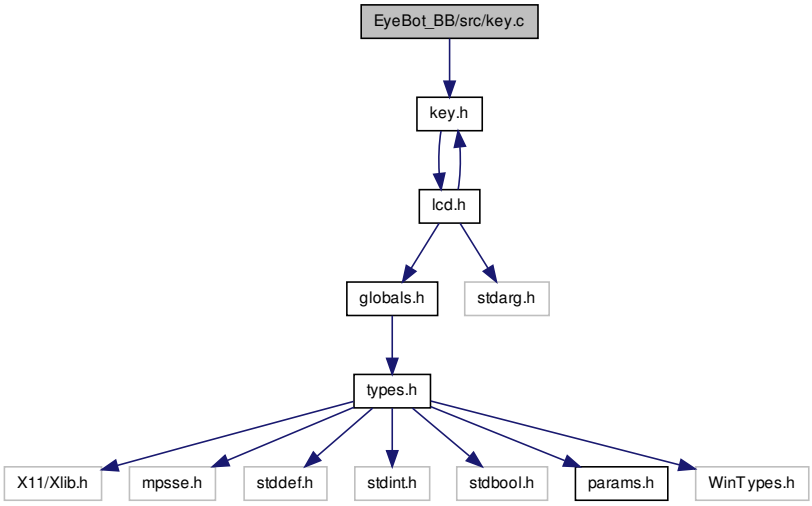
int retVal : Next code from the buffer

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_REGIONS 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)

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_t	rect : rectangle structure

Returns

int retVal : non-null if inside

5.26.2.2 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_t**	rawtouch : pointer to touch_event_t structure

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_t**	rawtouch : pointer to touch_event_t structure

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_t**	region : Pointer to a storage for region data

Returns

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_t***	ptouch_map : Pointer to a touch_map_t structure

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_t**	rawtouch : pointer to touch_event_t structure this is optional! only if raw 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-_t**	region : Pointer to a region data

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.

Parameters

keycode_t	excode : Expected keycode values (bit XORed)

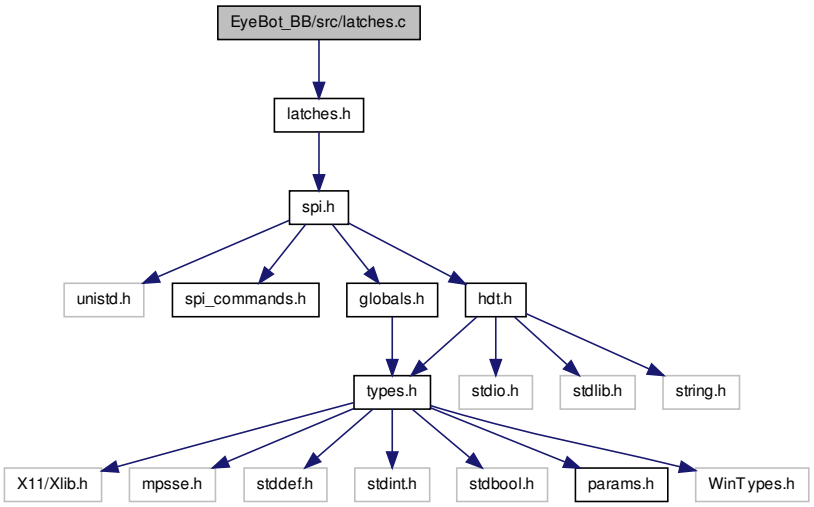
Returns

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.

Returns

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

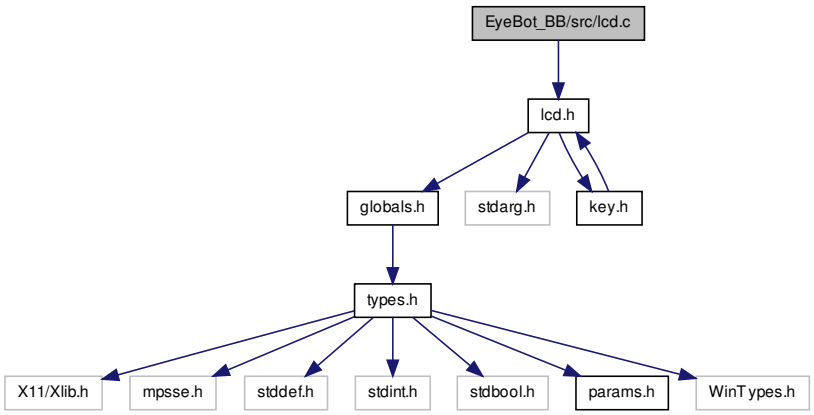
Returns

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_SHOWMENU flag and refresh the LCD.
- int LCDMenuI (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)

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 LCDInit() 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.

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

5.28.2.4 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.

Returns

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 LCDInit() for textsize.

Parameters

*fbinfo_t**	pinfo : Pointer to storage for screen & cursor info

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
*int**	column : Pointer to cursor column index

Returns

int retVal : always 0

5.28.2.15 int LCDInit ( )

Initialize the LCD.

Returns

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.

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

Returns

int retVal : always 0

5.28.2.19 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.

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.

Returns

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

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.

Parameters

*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

Returns

int retVal : always 0

5.28.2.27 int LCDMenuI (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

int	index : position of the menuitem

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

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

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

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).

Parameters

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

Returns

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
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

5.28.2.37 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

hword_t	mode : Mode flag - bit XORed

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.

Parameters

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

Returns

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

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).

Parameters

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

Returns

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).

Parameters

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

Returns

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

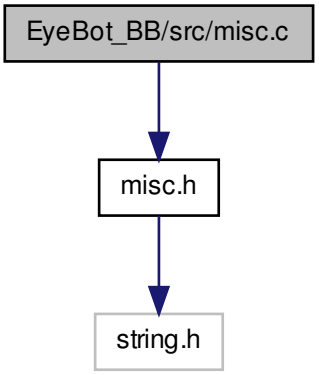
Returns

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.

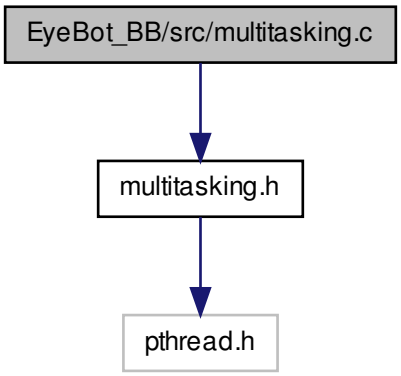
Author

Remi KEAT

5.30 EyeBot_BB/src/multitasking.c File Reference

Defines multitasking functions.

`#include "multitasking.h"` Include dependency graph for multitasking.c:



5.30.1 Detailed Description

Defines multitasking functions.

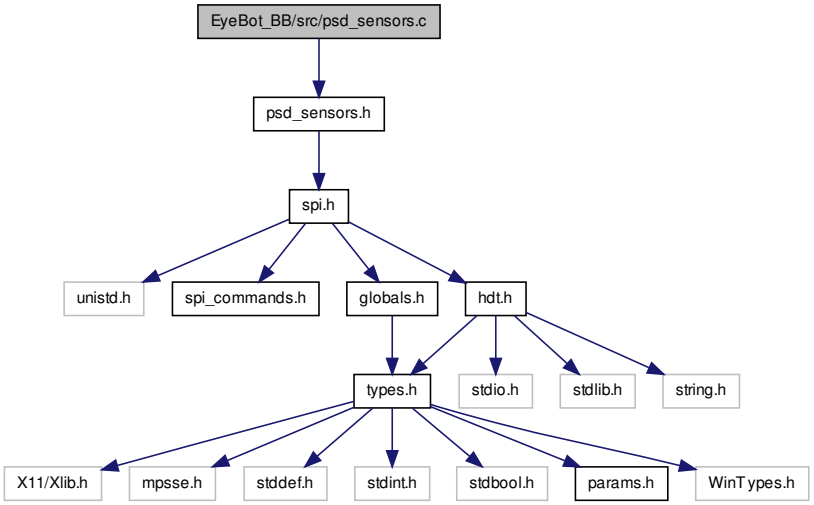
Author

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

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	semantics : unique definition for desired PSD

Returns

PSDHandle psdHandle : unique handle for all further operations

5.31.2.4 int PSDRelease (PSDHandle *psd*)

Stops measurings and releases a PSD.

Parameters

PSDHandle	psd

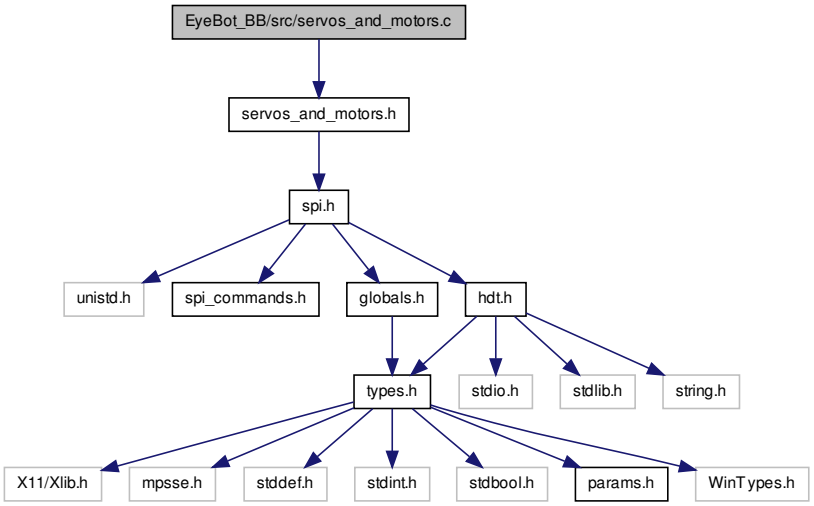
Returns

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	handle : ONE decoder-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)

Parameters

Device-Semantics	semantics

Returns

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	handle : ONE decoder-handle

Returns

long value of the encoder

5.32.2.7 int QUADRelease (QUADHandle *handle*)

Release one or more Quadrature-Decoder.

Parameters

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

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

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

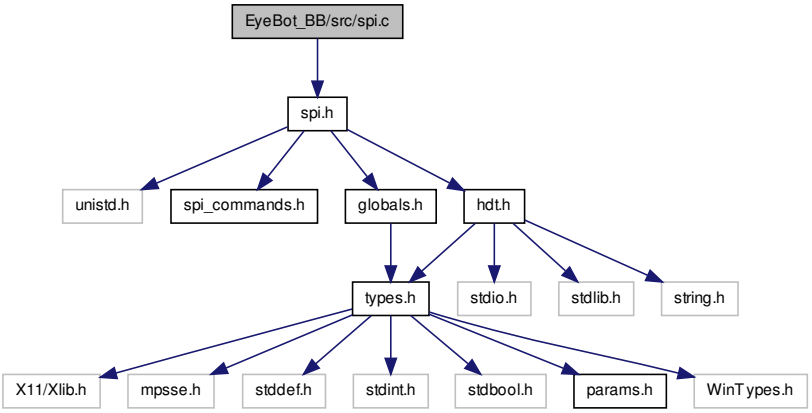
Returns

int retVal : always 0

5.33 EyeBot_BB/src/spi.c File Reference

Defines fonctions for sending and receiving SPI messages.

`#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.

Parameters

size_t	length
const	uint8_t data[]

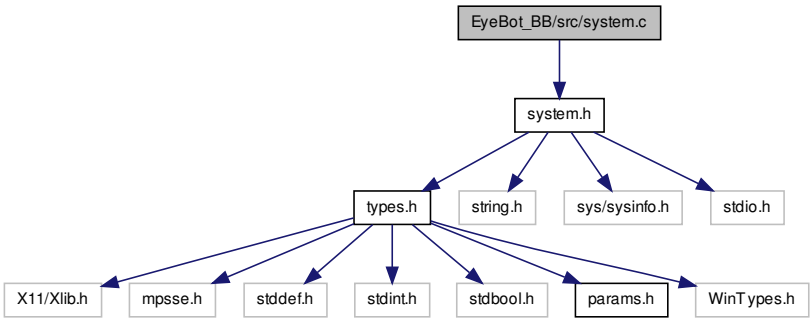
Returns

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 memory 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

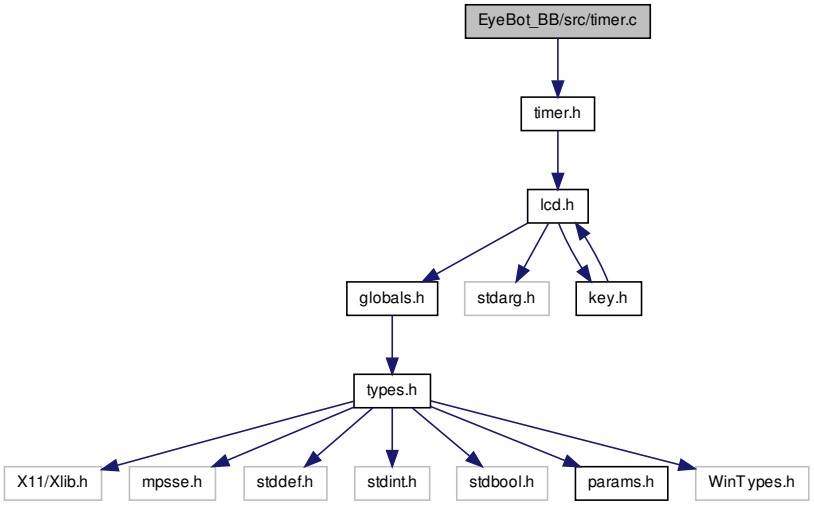
char* version : OS version

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.

Parameters

int	n : time to wait

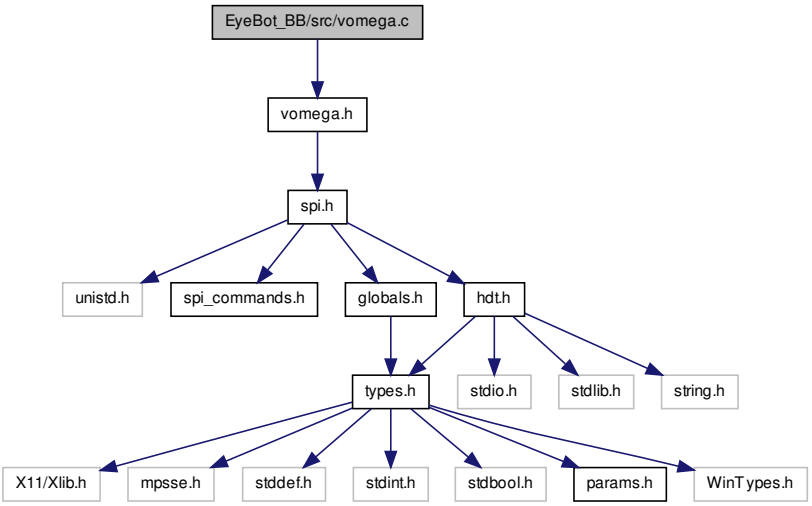
Returns

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, 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.

5.36.1 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
meterPerSec	v : speed to drive with (always positive!)

delta :

- pos. -> forward
- neg. -> 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
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

5.36.2.3 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