Smarties2

0.9

Generated by Doxygen 1.5.5

Fri Aug 29 13:48:09 2008

Contents

Chapter 1

Smarties2 software

1.1 License

GPL2 Licence

1.2 Description

This is the Source code's API and flow documentation for Smarties2

1.3 Architecture

Goal of this application is a state machine controlled by status flags. The main() function controls the program flow by reading and setting status flags. These status flags are polled each millisecond in an timer interrupt routine. This timer interrupt routine reads and sets the IO ports and sets corresponding status flags.

Following image clarifies the structure of the software:

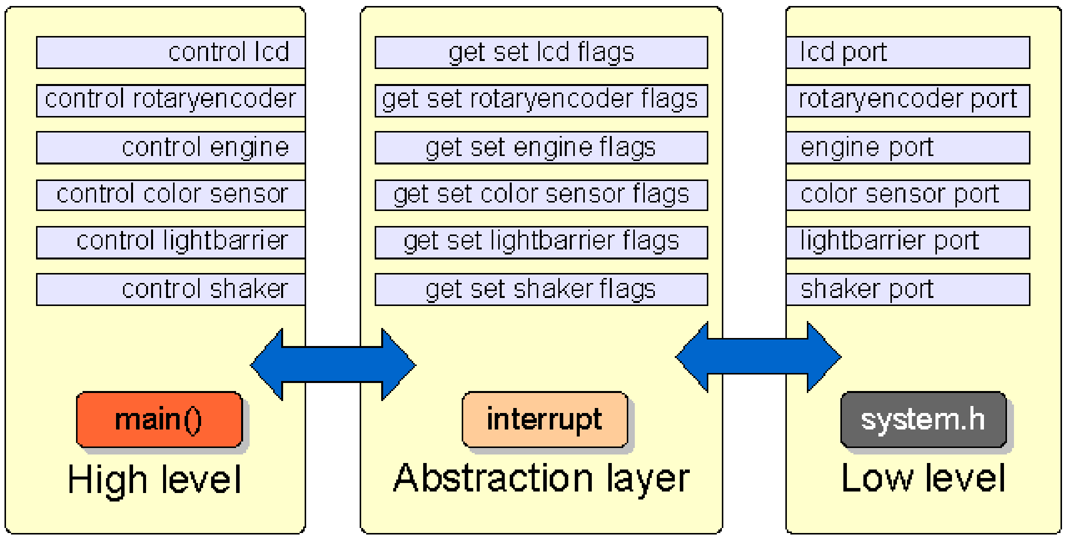


Figure 1.1: Layers of the software stacke

The task of the differen layers, high level, abstraction layer and low level can be described like in following image:

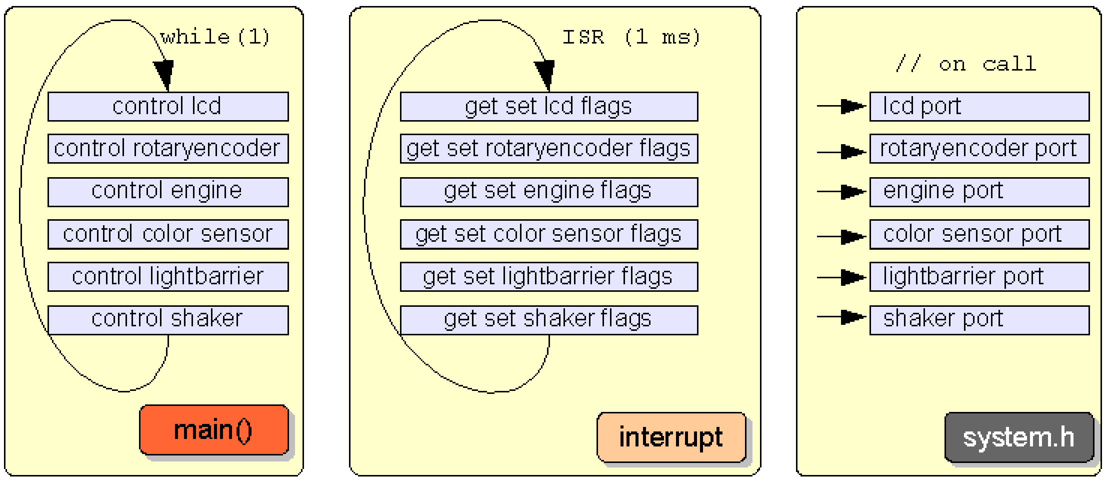


Figure 1.2: Executing the different layers

The state machine is devided into two sections, the mode and steps. Modes are represented as an enum system_mode_t and the steps as an struct system_step_t

The modes are changed depending on the user inputs or after powering on/reseting. The next picture clarifies the modes of the state machine.

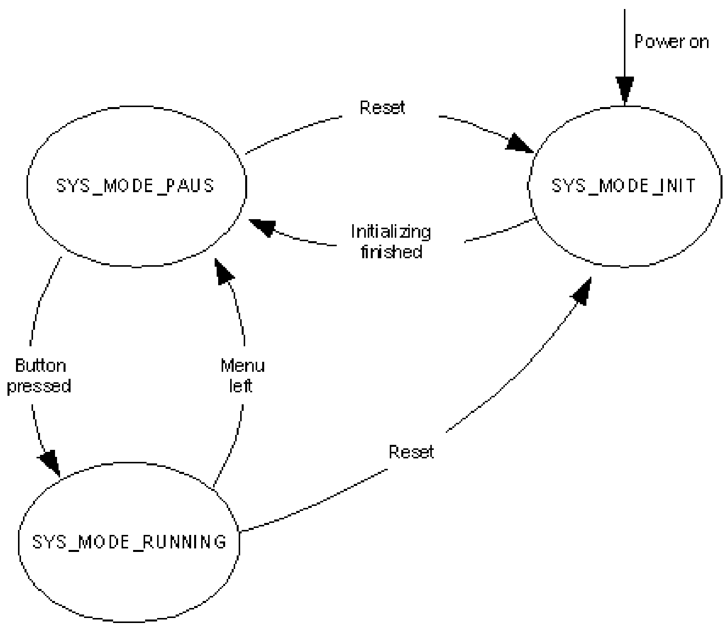


Figure 1.3: State diagram of the smartie sorter

The mode SYS_MODE_RUNNING equals the automatic mode, where everything is controlled in several steps. The last step, step III, is a transition step to begin from the start again. See next picture for the overview of the steps.

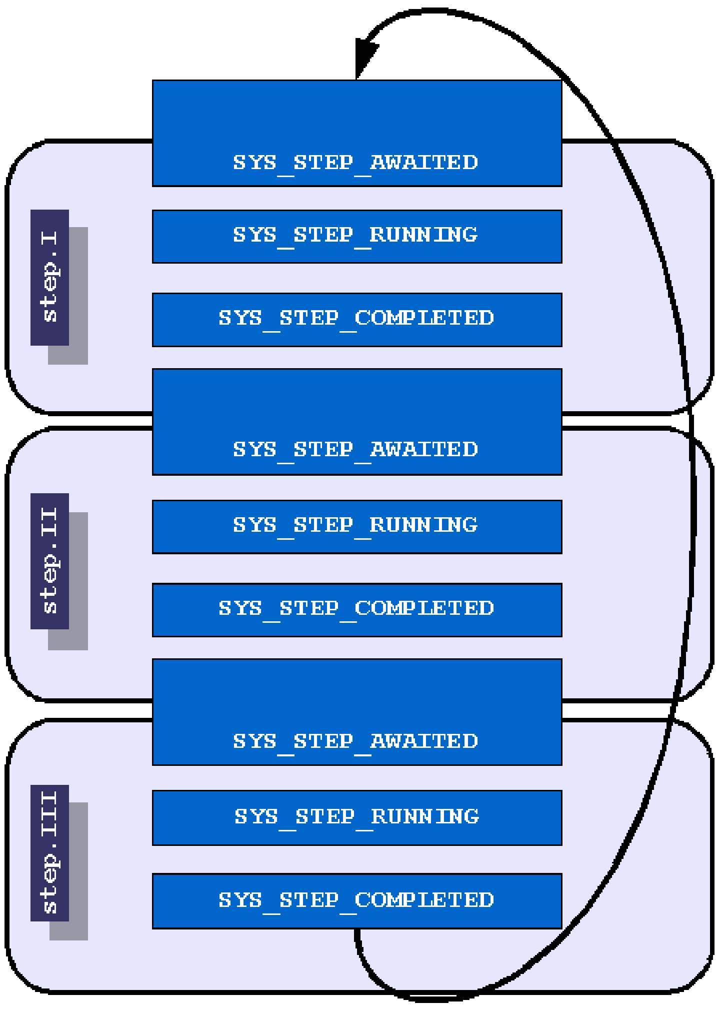


Figure 1.4: Executing steps of the mode SYS_MODE_RUNNING

Each steps starts several tasks and waits until they are finished. Then the next step will be entered.

To see what is happening exactly in the different steps, please have a look at the sourcecode.

The modes, steps and all input/output related parts of the Smartie sorter are administrated within structs. The structs are organized like in the following picture.

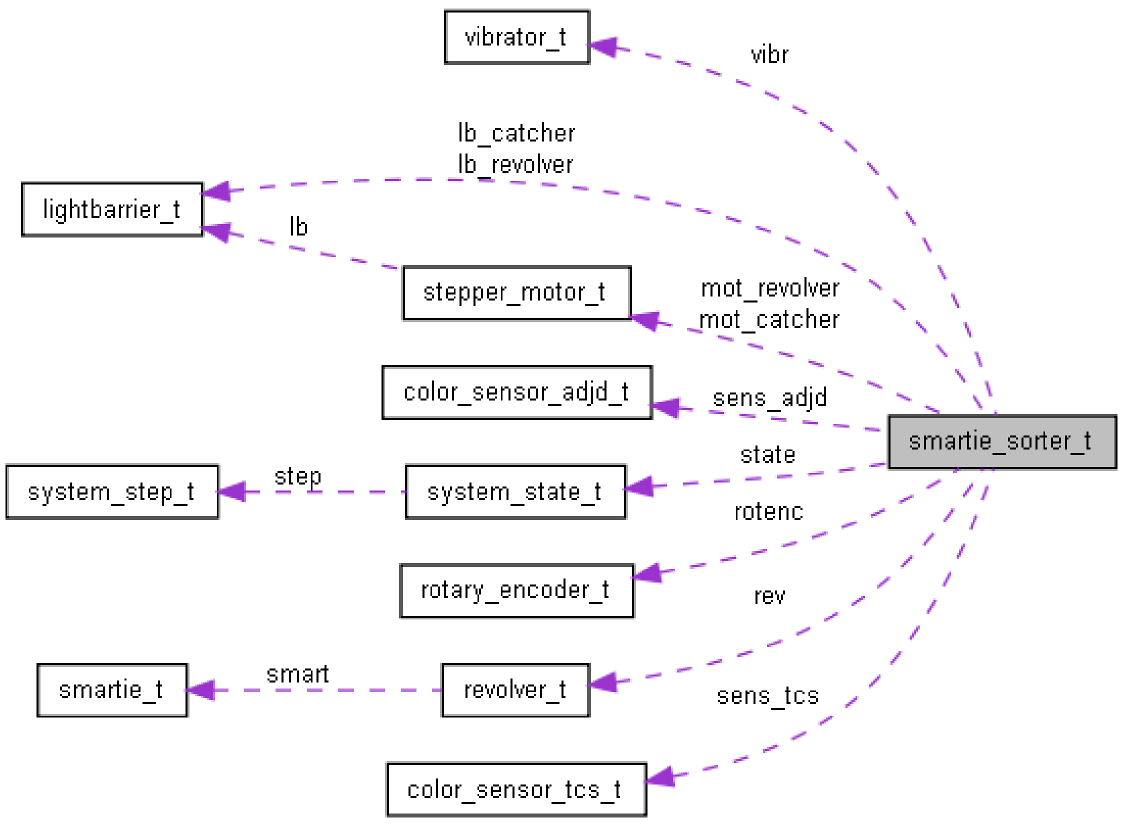


Figure 1.5: Brief overview of several objects (elements) of the smartie sorter

For the detailed overview and description please refere to the code and documentation of smartie_sorter_t. The system related IO actions are all defined in system.h There are controlled

- moving the revolver
- moving the catcher
- user input controlls

Minor configurations are made in smarties2.h

1.4 Progam flow

The application entry point is located at main() in smarties2.c file.

- The main function first performs the initialization
- It handles the modes of the smartie sorter
- It handles the state machine
- It handles the programs executed by the menu during SYS_MODE_PAUSE

The LCD controlling is done with the lcd_display.h

The Menu structure is described in menu.h

1.5 Color detection

The color sensor TCS230 delivers 4 output values:

- Blue (with blue filter)
- Green (with green filter)
- Red (with red filter)
- Brigthness (with no filter)

The smartie color detection is done by calculating the smallest distance to a next smartie.

For reference measures some values for each channel are recorded to gain a avarage value. They can be represented in an 3 Dimensional graph. For color detection the avarage value for each channel is used.

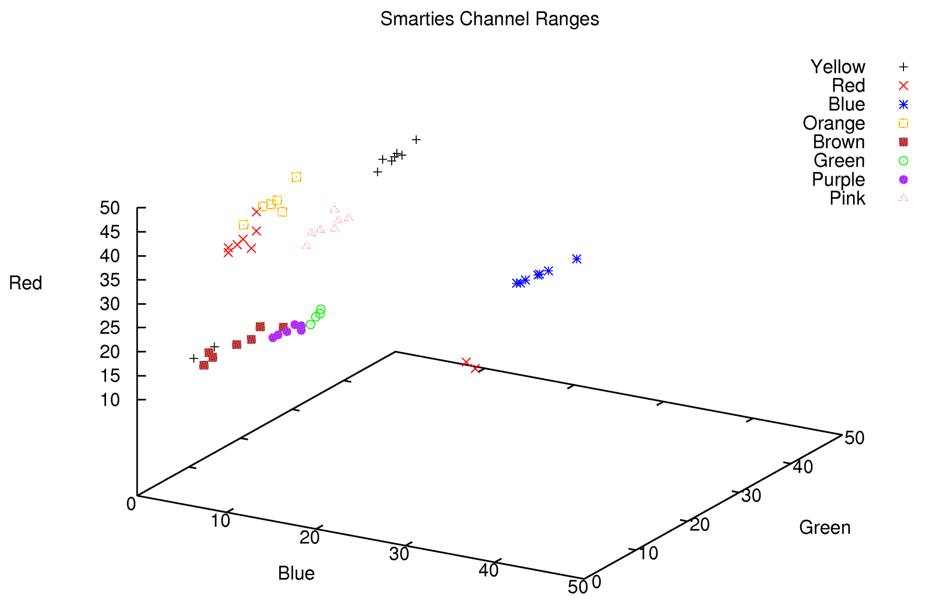


Figure 1.6: Smartie color RGB valus

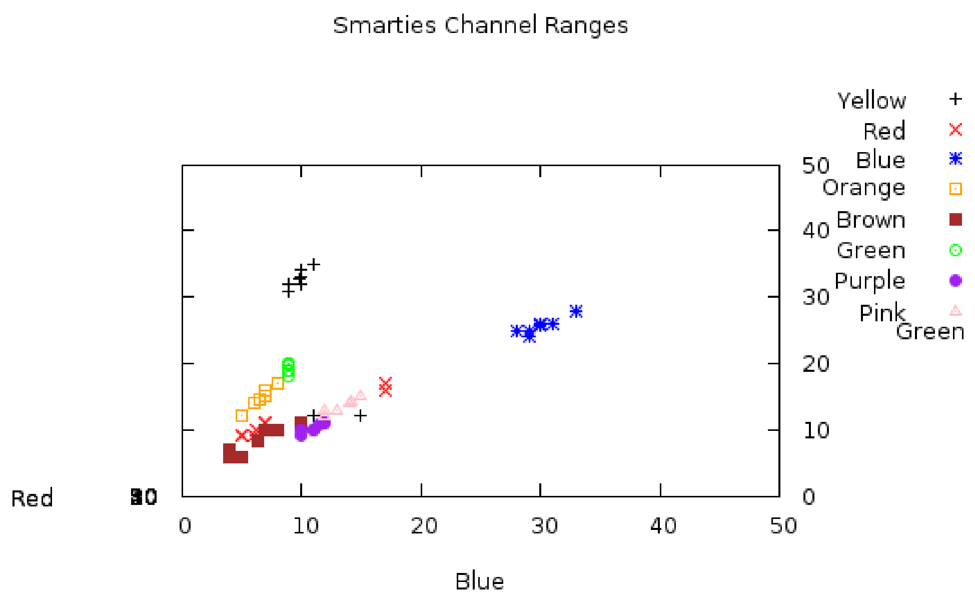


Figure 1.7: Smartie color RGB valus from top

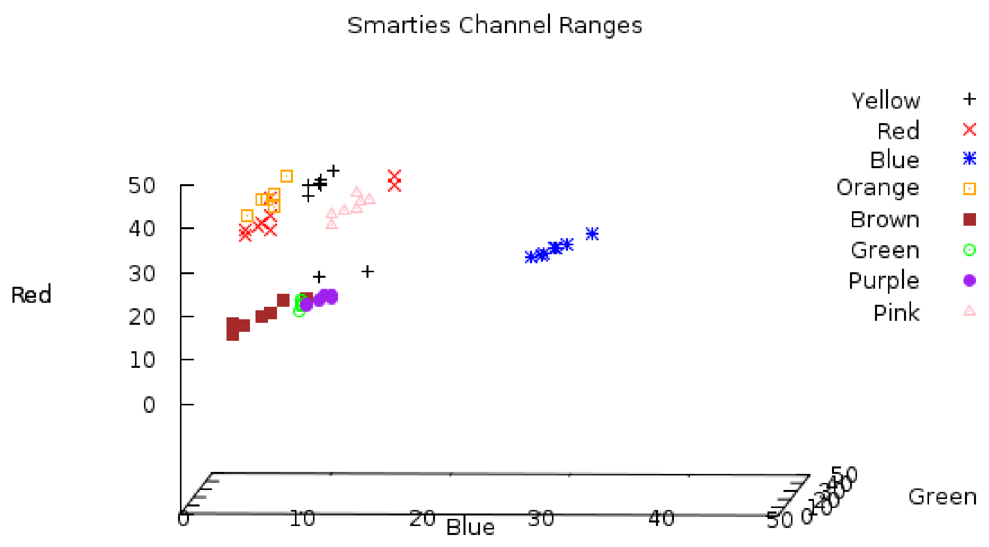


Figure 1.8: Smartie color RGB valus from front

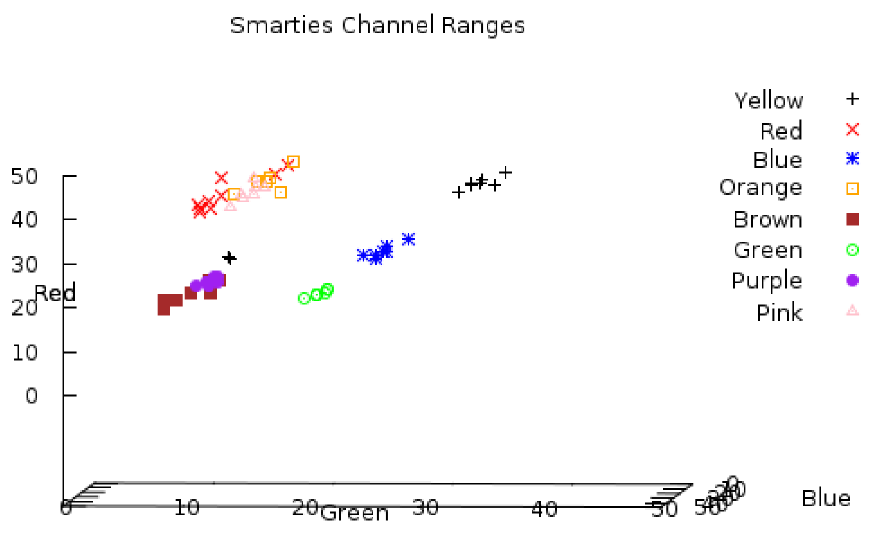


Figure 1.9: Smartie color RGB valus from side

If a smartie color's red, green and blue cannel are measured the distance to each reference smartie is worked out by following formula:

$$Distance = \sqrt{(blue_{new} - blue_{ava})^2 + (green_{new} - green_{ava})^2 + (red_{new} - red_{ava})^2}$$

The smartie sorter uses reference values which are gained during this software development. However the user can callibrate the reference values new without destroying the system default values.

The color tables with avarage values are stored in the EEprom memory. In the EEProm memory there are stored system default values as well as newly callibrated values. When the system default values are restored, all callibrated colors are overwritten.

Only the blue, green and red channel is respected. A survey brought up that the brightness of the surrounding has no influence to the color measurent. The most important factor is temperature, as the smarties fat drifts out when they are getting to warm (above 24 Deg C). Then, the smartie's colors become brighter.

1.6 Advanced color detecection

The first try with color detectin was made with reference tables. For each smartie was a minimum and maximum value for each channel stored. However, this method was too unrelyable.

More methods for calculating the correct smartie color are prepared in the code. They can be enabled by compiler switches. Enabling all methods could possible fill all data memory, as a lot of reference data is necessary, which are preferable stored as floats.

Another try was to calculate the normalized distance from the new, unknown smartie to the reference values. However this method didn't show good results, probably because the Orange, Red, Pink and Brown smarties are nearly all on a vector in one direction (see figure above or the 3D gnuplot).

Another idea (not implemented) for color detecting was respecting the color drift of smarties with the temperature. To respect this, smarties must be mesured in a temperature range from 15 Deg C to 25 Deg C and measure the unfilterd color channel (Brightness). Then, estimate the polynomial function of the three channels blue, green, red with the brightness as coefficient. The result is a curve in a 3 Dimensional space for each smartie color, and the axes are the three color channels. The next step is to calculate the orthogonal distance from a new, unknown smartie to all the curves. The curve which has the smallest orthogonal distance should belong to the corresponding smartie.

With this method it could also be possible to estimate the temperature of the surrounding, not with +-1 Deg C, but you could say that the surrounding is cold, warm or too warm. Maybe too warm for smarties, which should be kept below 25 Deg C.

1.7 Programs

During SYS_MODE_PAUSE various programs can be started from the menu. For controlling the state machines programs, the enum program_t is used.

Usually programs are executed completely in the background and only their progress or results are displayed during the state machine proceeds.

However, Programs can also take control over the user inputs and display. Some programs need to be completely finished before the state machine may proceed (e. g. prog_set_colors_blue)

The state of programs are controlled in main()

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 color_sensor_adjd_t Struct Reference

Describes the ADJD-S371 color sensor.

```
#include <system.h>
```

Data Fields

- common_stat status

  The current status of the color sensor.

- common_stat status_last

  The last status of the color sensor.

- smartie_color color

  The value from the last color detection.

4.1.1 Detailed Description

Describes the ADJD-S371 color sensor.

Definition at line 396 of file system.h.

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

- system.h

4.2 color_sensor_tcs_t Struct Reference

Describes the TCS230 color sensor.

```
#include <system.h>
```

Data Fields

- common_stat status

  The current status of the color sensor.

- common_stat status_last

  The status before current status.

- smartie_color color

  The value from the last color detection.

- int16_t time

  Especially for the TCS frequency mesurement. In milliseconds.

- int16_t filter_freq_blue

  The clock frequency in kHz measured with blue filter on.

- int16_t filter_freq_green

  The clock frequency in kHz measured with green filter on.

- int16_t filter_freq_red

  The clock frequency in kHz measured with red filter on.

- int16_t filter_freq_none

  The clock frequency in kHz measured with no filter on. Could be interpreted as general brightness.

- int16_t slopes

  The amount of slopes recognised during COL_SENS_TCS_SAMPLE_TIME.

- int16_t distance

  Needed for the distance to an reference value.

4.2.1 Detailed Description

Describes the TCS230 color sensor.

Definition at line 405 of file system.h.

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

- system.h

4.3 ee_memory_t Struct Reference

Structure in EEProm.

```
#include <ee.h>
```

Data Fields

- uint8_t dummy

  Dummy! Don't use!

- color_avarage def_blu

  System default color reference values for all smarties, blue channel.

- color_avarage def_gre

  System default color reference values for all smarties, green channel.

- color_avarage def_red

  System default color reference values for all smarties, bred channel.

- color_avarage usr_blu

  User calibrated color reference values for all smarites, blue channel.

- color_avarage usr_gre

  User calibrated color reference values for all smarites, green channel.

- color_avarage usr_red

  User calibrated color reference values for all smarites, red channel.

- uint16_t speed

  Smartie sorter speed , see smartie_sorter_t.

4.3.1 Detailed Description

Structure in EEProm.

Important! For keeping backward compatibility add new variables only at the end!

Definition at line 16 of file ee.h.

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

- ee.h

4.4 lightbarrier_t Struct Reference

Describes the module lightbarrier.

`#include <system.h>`

Data Fields

- lightbarrier_status status

  The actual status of the lightbarrier.

- lightbarrier_status status_last

  For recognising a pass.

- uint8_t passes

  The amount of recognized passes through the lightbarrier.

- int8_t(∗ is_blocked)()

  Returns TRUE if the lightbarrier is blocked.

4.4.1 Detailed Description

Describes the module lightbarrier.

Definition at line 362 of file system.h.

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

- system.h

4.5 menu_entry_t Struct Reference

The menu structure.

```
#include <menu.h>
```

Data Fields

- void(∗ function)(void)

  If push button pressed, this function will be executed (if available).

- void(∗ l_action)(void)

  If rotary_encoder_t is rotated to left, this functino will be executed (if available).

- void(∗ r_action)(void)

  If rotary_encoder_t is rotated to right, this functino will be executed (if available).

- char ∗ text [2]

  Text on Display (Max 24 Characters, 2 lines).

- void ∗ topmenu

  The menu item above of current menu item.

- void ∗ submenu

  The menu item below of current menu item.

- void ∗ prev

  Previous Menu item.

- void ∗ next

  Next menu item.

4.5.1 Detailed Description

The menu structure.

Each menu entry stores a menu entry which is next (right) or previous (left) from itself. It also stores menu entries below (submenu) or menu entries above (topmenu) itself. Furthermore, each menu entry has a specific task wich is stored behind the function pointer. The 'task' is, for example, changing into the topmenu (go Back), or rotate the catcher.

There can also be functions be related to the left and right action. Before executing any function from the menu, the function pointer must be checked it is NULL or not!

The menu must be initialized, the memory must be reserved before the whole stucture can be used. This is done init_menu().

Definition at line 179 of file menu.h.

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

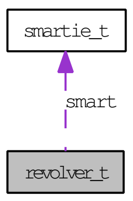
- menu.h

4.6 revolver_t Struct Reference

Describes the revolver module (the disc).

`#include <system.h>`

Collaboration diagram for revolver_t:



Data Fields

- smartie smart [REV_MAX_SIZE]

  A list of all smarties the revolver contains.

4.6.1 Detailed Description

Describes the revolver module (the disc).

Definition at line 431 of file system.h.

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

- system.h

4.7 rotary_encoder_t Struct Reference

The rotary encoder (user input device) structure.

```
#include <system.h>
```

Data Fields

- uint8_t push

  The amount of detected pushes.

- uint8_t right

  The amount of detected right turns.

- uint8_t left

  The amount of detected left turns.

- uint8_t pushtmp

  Stores temporarely status of pushbutton.

- uint8_t rottmp

  Stores temporarely status of rotation.

4.7.1 Detailed Description

The rotary encoder (user input device) structure.

Definition at line 313 of file system.h.

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

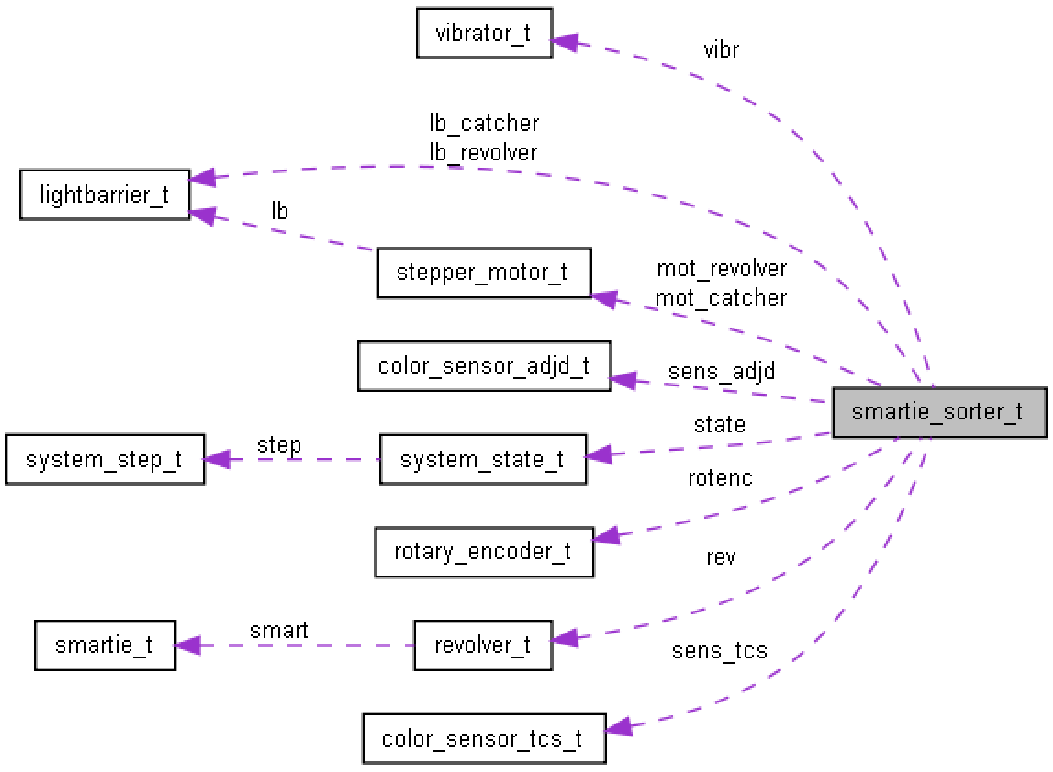
- system.h

4.8 smartie_sorter_t Struct Reference

All devices from the smartie sorter collected to one bundle.

```
#include <system.h>
```

Collaboration diagram for smartie_sorter_t:



Data Fields

- system_state state

  Stores the current state.

- program prog

  The programm which is running during SYS_MODE_PAUSE.

- color_sensor_adjd sens_adjd

  Digital color sensor.

- color_sensor_tcs sens_tcs

  Analog color sensor.

- stepper_motor mot_catcher

  Stepper motor for the catcher area.

- stepper_motor mot_revolver

Stepper motor for the revolver.

- lightbarrier lb_catcher

  Lightbarrier for the catcher.

- lightbarrier lb_revolver

  Lightbarrier for the revolver.

- vibrator vibr

  Shaker (or vibrator).

- rotary_encoder rotenc

  The rotary encoder (user input).

- revolver rev

  The revolver disc with smarties.

- uint16_t speed

  The operating speed (pause durations) the higher, the slower. See also sys_set_speed().

4.8.1 Detailed Description

All devices from the smartie sorter collected to one bundle.

Definition at line 439 of file system.h.

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

- system.h

4.9 smartie_t Struct Reference

Describes the properties a smartie, which is transported by the revolver_t, can have.

```
#include <system.h>
```

Data Fields

- smartie_color color

  Merged color.

- smartie_color color1

  From TCS color sensor.

- smartie_color color2

  From ADJD color sensor.

4.9.1 Detailed Description

Describes the properties a smartie, which is transported by the revolver_t, can have.

Definition at line 344 of file system.h.

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

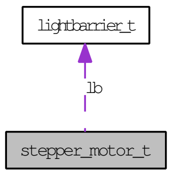
- system.h

4.10 stepper_motor_t Struct Reference

The stepper motor.

`#include <system.h>`

Collaboration diagram for stepper_motor_t:



Data Fields

- common_stat status

  Status.

- common_stat status_last

  The last status.

- int8_t current_pos

  Current position (smartie_color_t can also be used).

- int8_t target_pos

  Target position (smartie_color_t can also be used).

- uint16_t cycle_counter

  One cycle takes 1 millisecond.

- uint8_t steps

  Count every step to estimate when to ramp down.

- uint8_t ramp_steps

  One step takes CATCH_STEP_DURATION or REV_STEP_DURATION steps.

- uint8_t ramp_duration

  Ramp duration lasts CATCH_RAMP_DURATION or REV_RAMP_DURATION.

- uint8_t steps_estimated

  Steps counted from as soon as the lightbarriere is blocked.

- uint8_t steps_estim_def

  Start (default) value for CATCH_STEPS_ESTIMATED or REV_STEPS_ESTIMATED.

- int8_t max_size

  Max. positions. CATCH_MAX_POS or REV_MAX_POS.

- int8_t step_duration

  The duration of REV_STEP_DURATION or CATCH_STEP_DURATION cycles lasts one step.

- int16_t pause

  The motor pauses as long as this is not zero.

- int16_t pause_duration

  Default value (CATCH_PAUSE_DURATION or REV_PAUSE_DURATION).

- lightbarrier ∗ lb

  The lightbarrier that corresponds to that motor.

- void(∗ enable)()

  Will enable the motor (power on).

- void(∗ disable)()

  Will diable the motor (power off).

- void(∗ move_step)()

  Will move the motor for one step.

4.10.1 Detailed Description

The stepper motor.

Definition at line 372 of file system.h.

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

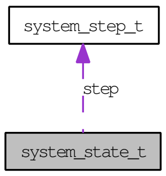
- system.h

4.11 system_state_t Struct Reference

Grouping of mode and steps.

```
#include <system.h>
```

Collaboration diagram for system_state_t:



Data Fields

- system_mode mode

  Stores the current mode.

- system_mode mode_last

  Stores the last mode for transition steps.

- system_step step

  Stores the current step of the mode.

4.11.1 Detailed Description

Grouping of mode and steps.

Definition at line 272 of file system.h.

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

- system.h

4.12 system_step_t Struct Reference

The single steps during running mode SYS_MODE_RUNNING.

```
#include <system.h>
```

Data Fields

- common_stat I

    detecting colors and position catcher

- common_stat II

    positioning revolver

- common_stat III

    begin new mode cycle

4.12.1 Detailed Description

The single steps during running mode SYS_MODE_RUNNING.

Definition at line 263 of file system.h.

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

- system.h

4.13 vibrator_t Struct Reference

Describes the virbrator (shaker) module.

```
#include <system.h>
```

Data Fields

- common_stat status

  Status.

- common_stat status_last

  The last status.

- uint16_t duration

  How long to vibrate the shaker in ms (SHAKER_DURATION).

4.13.1 Detailed Description

Describes the virbrator (shaker) module.

Definition at line 421 of file system.h.

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

- system.h

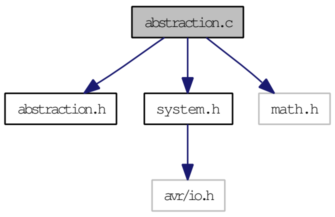
Chapter 5

File Documentation

5.1 abstraction.c File Reference

```
#include "abstraction.h"
#include "system.h"
#include <math.h>
```

Include dependency graph for abstraction.c:



Functions

- void motor_stuff ()

  Takes controll over the catcher and revolver stepper engines.

- void rotary_encoder_stuff ()

  Takes controll over the rotary encoder (user input) device.

- void lightbarrier_stuff ()

  Take control over the lightbarriers.

- void sensor_tcs_stuff ()

  Takes control over the TCS color sensor.

- void vibrator_stuff ()

  Takes control over the shaker (vibrator) device.

5.1.1 Detailed Description

All these functions are expected to be called every millisecond by the function ISR (TIMER0_OVF_vect) in the file my_interrupt.c

Definition in file abstraction.c.

5.1.2 Function Documentation

5.1.2.1 void lightbarrier_stuff ()

Take control over the lightbarriers.

This function is expected to be called regulary. It polls the input pins of the lightbarrier and sets the corresponding flags of lightbarrier struct. This function assumes that lightbarriers are always enabled.

Definition at line 236 of file abstraction.c.

References IS_LB_CATCHER, IS_LB_REVOLVER, lb_blocked, smartie_sorter_t::lb_catcher, lb_free, smartie_sorter_t::lb_revolver, lightbarrier_t::passes, lightbarrier_t::status, and lightbarrier_t::status_last.

Referenced by ISR().

5.1.2.2 void motor_stuff ()

Takes controll over the catcher and revolver stepper engines.

The ramp up is made by linear shrinking the time period t for each step:

```
 Steps
 ^
 |         4t                 3t        2t   t  t  t  t
 *                       *           *     *  *  *  *  *
 |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-> Time
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18

Symbolic diagram, no real values
```

This function is expected to be called every millisecond to work properly.

Important setting values are:

- CATCH_STEP_DURATION
- CATCH_RAMP_DURATION
- REV_STEP_DURATION
- REV_RAMP_DURATION

Definition at line 68 of file abstraction.c.

References smartie_sorter_t::mot_catcher, and smartie_sorter_t::mot_revolver.

Referenced by ISR().

5.1.2.3 void rotary_encoder_stuff ()

Takes controll over the rotary encoder (user input) device.

This function is expected to be called regulary. It polls the input pins for the rotary encoder and sets the corresponding flags of the rotary_encoder struct.

Definition at line 170 of file abstraction.c.

References IS_ROTENC_A, IS_ROTENC_AB, IS_ROTENC_B, IS_ROTENC_NONE, IS_ROTENC_-PUSH, rotary_encoder_t::left, rotary_encoder_t::push, rotary_encoder_t::pushtmp, rotary_encoder_-t::right, smartie_sorter_t::rotenc, ROTENC_A, ROTENC_B, ROTENC_BOTH, ROTENC_NONE, ROTENC_PUSH, and rotary_encoder_t::rottmp.

Referenced by ISR().

5.1.2.4 void sensor_tcs_stuff ()

Takes control over the TCS color sensor.

This function is expected to be called exectly every millisecond to work properly. This function will read the current color of the smartie.

About the way of color detection please refere to the Main page

Definition at line 283 of file abstraction.c.

References col_blue, col_green, col_red, COL_SENS_TCS_DISABLE, COL_SENS_TCS_ENABLE, COL_SENS_TCS_FREQ_MESURE_DI, COL_SENS_TCS_FREQ_MESURE_EN, COL_SENS_TCS_-SAMPLE_TIME, COL_SENS_TCS_SET_FILTER, col_unknown, color_sensor_tcs_t::color, color_-sensor_tcs_t::distance, color_sensor_tcs_t::filter_freq_blue, color_sensor_tcs_t::filter_freq_green, color_-sensor_tcs_t::filter_freq_none, color_sensor_tcs_t::filter_freq_red, smartie_sorter_t::sens_tcs, color_-sensor_tcs_t::slopes, stat_finished, stat_idle, stat_start_working, stat_stop_working, stat_working, color_-sensor_tcs_t::status, color_sensor_tcs_t::status_last, and color_sensor_tcs_t::time.

Referenced by ISR().

5.1.2.5 void vibrator_stuff ()

Takes control over the shaker (vibrator) device.

This function is expected to be called regulary. It polls the input pins of the shaker and sets the corresponding flags of the shaker struct. This function assumes that

Definition at line 428 of file abstraction.c.

References vibrator_t::duration, stat_finished, stat_idle, stat_start_working, stat_stop_working, stat_-working, vibrator_t::status, vibrator_t::status_last, smartie_sorter_t::vibr, VIBR_DURATION, VIBR_-OFF, and VIBR_ON.

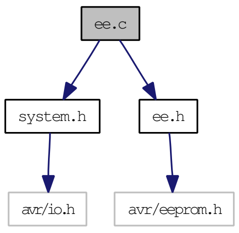
Referenced by ISR().

5.2 ee.c File Reference

Values for system defaults, stored in EEProm.

```
#include "system.h"
#include "ee.h"
```

Include dependency graph for ee.c:



Variables

- const ee_memory ee_mem EEMEM

  Values for structure in EEProm.

5.2.1 Detailed Description

Values for system defaults, stored in EEProm.

When compiling this file, a smarties2.eep file int intel hex format will be created, which can be uploaded to the EEprom of the ATMega32. The ATMega32 will not override the EEProm with this this values on reset.

Definition in file ee.c.

5.2.2 Variable Documentation

5.2.2.1 const ee_memory ee_mem EEMEM

Values for structure in EEProm.

Important! For keeping backward compatibility add new variables only at the end!

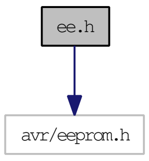
Definition at line 26 of file ee.c.

5.3 ee.h File Reference

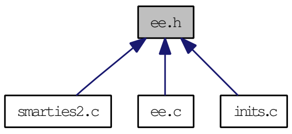
Some EEProm memory administration.

```
#include <avr/eeprom.h>
```

Include dependency graph for ee.h:



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



Data Structures

- struct ee_memory_t

    Structure in EEProm.

5.3.1 Detailed Description

Some EEProm memory administration.

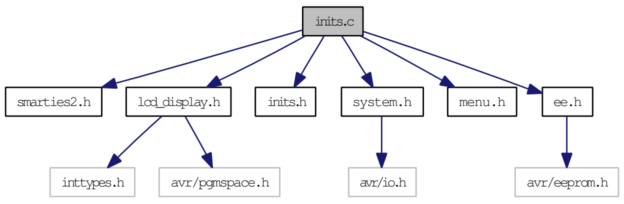
Definition in file ee.h.

5.4 inits.c File Reference

Some init functions.

```
#include "smarties2.h"
#include "lcd_display.h"
#include "inits.h"
#include "system.h"
#include "menu.h"
#include "ee.h"
```

Include dependency graph for inits.c:



Functions

- void init_all ()

  Calls all necessary init functions.

- void init_io ()

  Configures nearly all General IO pins, disables JTAG.

- void init_sensor_tcs ()

  Set up the TCS color sensor.

- void init_timer ()

  This will generate an Interrupt every millisecond by Timer 0 on compare match.

- void init_memory ()

  Inits important system variables, also from EEprom.

- void init_motors ()

  Brings revolver and catcher to defined positions; Set up important motor values.

- void init_menu ()

  Creates the menu stucture, connects menus, connects functions to menus.

5.4.1 Detailed Description

Some init functions.

Copyright (c) 2008 Simeon Felis

Definition in file inits.c.

5.4.2 Function Documentation

5.4.2.1 void init_menu ()

Creates the menu stucture, connects menus, connects functions to menus.

The menu structure and functionality is explained in menu.h in detailed.

Definition at line 228 of file inits.c.

References menu_entry_t::function, menu_entry_t::l_action, menu_entry_t::next, menu_entry_t::prev, menu_entry_t::r_action, menu_entry_t::submenu, sys_catcher_rotate(), sys_enter_submenu(), sys_enter_-topmenu(), sys_measure_tcs(), sys_pause(), sys_reference_measure_blue(), sys_reference_measure_-brown(), sys_reference_measure_green(), sys_reference_measure_orange(), sys_reference_measure_-pink(), sys_reference_measure_purple(), sys_reference_measure_red(), sys_reference_measure_restore(), sys_reference_measure_yellow(), sys_resume(), sys_revolver_rotate(), sys_speed_down(), sys_speed_-up(), menu_entry_t::text, and menu_entry_t::topmenu.

Referenced by init_all().

5.4.2.2 void init_timer ()

This will generate an Interrupt every millisecond by Timer 0 on compare match.

The interrupt settings are related like following:

$$T_{CompareMatch} = (F_{CPU})^{-1} \cdot Prescaler \cdot (Register_{OutputCompare})$$

There is an error of about 0.8% from 1 Millisecond

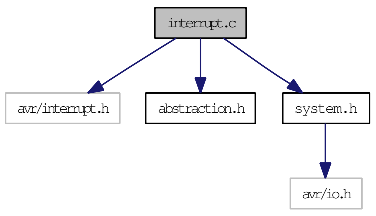
Definition at line 123 of file inits.c.

Referenced by init_all().

5.5 interrupt.c File Reference

```
#include <avr/interrupt.h>
#include "abstraction.h"
#include "system.h"
```

Include dependency graph for interrupt.c:



Functions

- ISR (TIMER0_COMP_vect)

  Interrupt routine, executed every millisecond.

- ISR (INT0_vect)

  Interupt routine for measuring frequency of color sensor TCS OUT pin.

5.5.1 Detailed Description

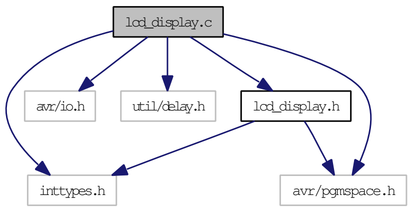
Alle necessary interrupt routines. Please check the source code of this file for more documentation.

Definition in file interrupt.c.

5.6 lcd_display.c File Reference

```
#include <inttypes.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include "lcd_display.h"
```

Include dependency graph for lcd_display.c:



Functions

- void lcd_command (uint8_t cmd)

    Send LCD controller instruction command.

- void lcd_data (uint8_t data)

    Send data byte to LCD controller.

- void lcd_gotoxy (uint8_t x, uint8_t y)

    Set cursor to specified position.

- void lcd_clrscr (void)

    Clear display and set cursor to home position.

- void lcd_home (void)

    Set cursor to home position.

- void lcd_putc (char c)

    Display character at current cursor position.

- void lcd_puts (const char ∗s)

    Display string without auto linefeed.

- void lcd_puts_p (const char ∗progmem_s)

    Display string from program memory without auto linefeed.

- void lcd_init (uint8_t dispAttr)

    Initialize display and select type of cursor.

5.6.1 Detailed Description

Title : HD44780U LCD library Author: Peter Fleury <pfleury@gmx.ch> http://jump.to/fleury File:

Id

lcd.c,v 1.14.2.1 2006/01/29 12:16:41 peter Exp

Software: AVR-GCC 3.3 Target: any AVR device, memory mapped mode only for AT90S4414/8515/Mega

DESCRIPTION Basic routines for interfacing a HD44780U-based text lcd display

Originally based on Volker Oth's lcd library, changed lcd_init(), added additional constants for lcd_-command(), added 4-bit I/O mode, improved and optimized code.

Library can be operated in memory mapped mode (LCD_IO_MODE=0) or in 4-bit IO port mode (LCD_-IO_MODE=1). 8-bit IO port mode not supported.

Memory mapped mode compatible with Kanda STK200, but supports also generation of R/W signal through A8 address line.

USAGE See the C include lcd.h file for a description of each function

Definition in file lcd_display.c.

5.6.2 Function Documentation

5.6.2.1 void lcd_clrscr (void)

Clear display and set cursor to home position.

Parameters:

void

Returns:

none

Definition at line 434 of file lcd_display.c.

References lcd_command().

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

5.6.2.2 void lcd_command (uint8_t *cmd*)

Send LCD controller instruction command.

Parameters:

cmd instruction to send to LCD controller, see HD44780 data sheet

Returns:

none

Definition at line 372 of file lcd_display.c.

Referenced by lcd_clrscr(), lcd_gotoxy(), lcd_home(), and lcd_init().

5.6.2.3 void lcd_data (uint8_t *data*)

Send data byte to LCD controller.

Similar to lcd_putc(), but without interpreting LF

Parameters:

data byte to send to LCD controller, see HD44780 data sheet

Returns:

none

Definition at line 384 of file lcd_display.c.

5.6.2.4 void lcd_gotoxy (uint8_t *x*, uint8_t *y*)

Set cursor to specified position.

Parameters:

x horizontal position
(0: left most position)

y vertical position
(0: first line)

Returns:

none

Definition at line 398 of file lcd_display.c.

References lcd_command(), LCD_START_LINE1, LCD_START_LINE2, LCD_START_LINE3, and LCD_START_LINE4.

Referenced by main().

5.6.2.5 void lcd_home (void)

Set cursor to home position.

Parameters:

void

Returns:

none

Definition at line 443 of file lcd_display.c.

References lcd_command().

5.6.2.6 void lcd_init (uint8_t *dispAttr*)

Initialize display and select type of cursor.

Parameters:

dispAttr **LCD_DISP_OFF** display off
LCD_DISP_ON display on, cursor off
LCD_DISP_ON_CURSOR display on, cursor on
LCD_DISP_ON_CURSOR_BLINK display on, cursor on flashing

Returns:

none

Definition at line 538 of file lcd_display.c.

References lcd_clrscr(), lcd_command(), LCD_DATA0_PIN, LCD_DATA0_PORT, LCD_DATA1_-PIN, LCD_DATA1_PORT, LCD_DATA2_PIN, LCD_DATA2_PORT, LCD_DATA3_PIN, LCD_DATA3_-PORT, LCD_E_PIN, LCD_E_PORT, LCD_RS_PIN, LCD_RS_PORT, LCD_RW_PIN, and LCD_RW_-PORT.

Referenced by init_all().

5.6.2.7 void lcd_putc (char *c*)

Display character at current cursor position.

Parameters:

c character to be displayed

Returns:

none

Definition at line 454 of file lcd_display.c.

References LCD_DISP_LENGTH, LCD_START_LINE1, LCD_START_LINE2, LCD_START_LINE3, and LCD_START_LINE4.

Referenced by lcd_puts(), and lcd_puts_p().

5.6.2.8 void lcd_puts (const char ∗ *s*)

Display string without auto linefeed.

Parameters:

s string to be displayed

Returns:

none

Definition at line 501 of file lcd_display.c.

References lcd_putc().

Referenced by init_all(), main(), and smartie_lcd_write_color().

5.6.2.9 void lcd_puts_p (const char ∗ *progmem_s*)

Display string from program memory without auto linefeed.

Parameters:

s string from program memory be be displayed

Returns:

none

See also:

lcd_puts_P

Definition at line 518 of file lcd_display.c.

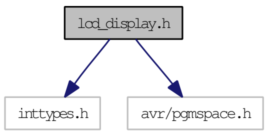
References lcd_putc().

5.7 lcd_display.h File Reference

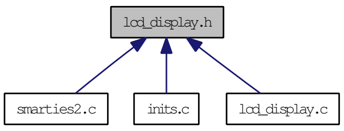
Basic routines for interfacing a HD44780U-based text LCD display.

```
#include <inttypes.h>
#include <avr/pgmspace.h>
```

Include dependency graph for lcd_display.h:



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



Functions

- #define lcd_puts_P(__s) lcd_puts_p(PSTR(__s))

  macros for automatically storing string constant in program memory

- void lcd_init (uint8_t dispAttr)

  Initialize display and select type of cursor.

- void lcd_clrscr (void)

  Clear display and set cursor to home position.

- void lcd_home (void)

  Set cursor to home position.

- void lcd_gotoxy (uint8_t x, uint8_t y)

  Set cursor to specified position.

- void lcd_putc (char c)

  Display character at current cursor position.

- void lcd_puts (const char ∗s)

  Display string without auto linefeed.

- void lcd_puts_p (const char ∗progmem_s)

  Display string from program memory without auto linefeed.

- void lcd_command (uint8_t cmd)

Send LCD controller instruction command.

- void lcd_data (uint8_t data)

  Send data byte to LCD controller.

Defines

Definitions for MCU Clock Frequency

Adapt the MCU clock frequency in Hz to your target.

- #define XTAL 16000000

Definition for LCD controller type

Use 0 for HD44780 controller, change to 1 for displays with KS0073 controller.

- #define LCD_CONTROLLER_KS0073 0

Definitions for Display Size

Change these definitions to adapt setting to your display

- #define LCD_LINES 2
- #define LCD_DISP_LENGTH 24
- #define LCD_LINE_LENGTH 0x40
- #define LCD_START_LINE1 0x00
- #define LCD_START_LINE2 0x40
- #define LCD_START_LINE3 0x14
- #define LCD_START_LINE4 0x54
- #define LCD_WRAP_LINES 0
- #define LCD_IO_MODE 1

Definitions for 4-bit IO mode

Change LCD_PORT if you want to use a different port for the LCD pins.

The four LCD data lines and the three control lines RS, RW, E can be on the same port or on different ports. Change LCD_RS_PORT, LCD_RW_PORT, LCD_E_PORT if you want the control lines on different ports.

Normally the four data lines should be mapped to bit 0..3 on one port, but it is possible to connect these data lines in different order or even on different ports by adapting the LCD_DATAx_PORT and LCD_DATAx_PIN definitions.

- #define LCD_PORT PORTC
- #define LCD_DATA0_PORT LCD_PORT
- #define LCD_DATA1_PORT LCD_PORT
- #define LCD_DATA2_PORT LCD_PORT
- #define LCD_DATA3_PORT LCD_PORT
- #define LCD_DATA0_PIN 2
- #define LCD_DATA1_PIN 3
- #define LCD_DATA2_PIN 4
- #define LCD_DATA3_PIN 5
- #define LCD_RS_PORT LCD_PORT
- #define LCD_RS_PIN 7
- #define LCD_RW_PORT PORTD
- #define LCD_RW_PIN 3
- #define LCD_E_PORT LCD_PORT

- #define LCD_E_PIN 6

Definitions for LCD command instructions

The constants define the various LCD controller instructions which can be passed to the function lcd_-command(), see HD44780 data sheet for a complete description.

- #define **LCD_CLR** 0
- #define **LCD_HOME** 1
- #define **LCD_ENTRY_MODE** 2
- #define **LCD_ENTRY_INC** 1
- #define **LCD_ENTRY_SHIFT** 0
- #define **LCD_ON** 3
- #define **LCD_ON_DISPLAY** 2
- #define **LCD_ON_CURSOR** 1
- #define **LCD_ON_BLINK** 0
- #define **LCD_MOVE** 4
- #define **LCD_MOVE_DISP** 3
- #define **LCD_MOVE_RIGHT** 2
- #define **LCD_FUNCTION** 5
- #define **LCD_FUNCTION_8BIT** 4
- #define **LCD_FUNCTION_2LINES** 3
- #define **LCD_FUNCTION_10DOTS** 2
- #define **LCD_CGRAM** 6
- #define **LCD_DDRAM** 7
- #define **LCD_BUSY** 7
- #define **LCD_ENTRY_DEC** 0x04
- #define **LCD_ENTRY_DEC_SHIFT** 0x05
- #define **LCD_ENTRY_INC_** 0x06
- #define **LCD_ENTRY_INC_SHIFT** 0x07
- #define **LCD_DISP_OFF** 0x08
- #define **LCD_DISP_ON** 0x0C
- #define **LCD_DISP_ON_BLINK** 0x0D
- #define **LCD_DISP_ON_CURSOR** 0x0E
- #define **LCD_DISP_ON_CURSOR_BLINK** 0x0F
- #define **LCD_MOVE_CURSOR_LEFT** 0x10
- #define **LCD_MOVE_CURSOR_RIGHT** 0x14
- #define **LCD_MOVE_DISP_LEFT** 0x18
- #define **LCD_MOVE_DISP_RIGHT** 0x1C
- #define **LCD_FUNCTION_4BIT_1LINE** 0x20
- #define **LCD_FUNCTION_4BIT_2LINES** 0x28
- #define **LCD_FUNCTION_8BIT_1LINE** 0x30
- #define **LCD_FUNCTION_8BIT_2LINES** 0x38
- #define **LCD_MODE_DEFAULT** ((1<<LCD_ENTRY_MODE) | (1<<LCD_ENTRY_INC))

5.7.1 Detailed Description

Basic routines for interfacing a HD44780U-based text LCD display.

5.7.2 License

```
Originally based on Volker Oth's LCD library,
changed lcd_init(), added additional constants for lcd_command(),
added 4-bit I/O mode, improved and optimized code.

Library can be operated in memory mapped mode (LCD_IO_MODE=0) or in
4-bit IO port mode (LCD_IO_MODE=1). 8-bit IO port mode not supported.

Memory mapped mode compatible with Kanda STK200, but supports also
generation of R/W signal through A8 address line.
```

Author:

Peter Fleury `pfleury@gmx.ch` `http://jump.to/fleury`

See also:

The chapter `Interfacing a HD44780 Based LCD to an AVR` on my home page.

Further documentation can be found in lcd_display.c

Definition in file lcd_display.h.

5.7.3 Define Documentation

5.7.3.1 #define LCD_CONTROLLER_KS0073 0

Use 0 for HD44780 controller, 1 for KS0073 controller

Definition at line 55 of file lcd_display.h.

5.7.3.2 #define LCD_DATA0_PIN 2

pin for 4bit data bit 0

Definition at line 92 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.3 #define LCD_DATA0_PORT LCD_PORT

port for 4bit data bit 0

Definition at line 88 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.4 #define LCD_DATA1_PIN 3

pin for 4bit data bit 1

Definition at line 93 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.5 #define LCD_DATA1_PORT LCD_PORT

port for 4bit data bit 1

Definition at line 89 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.6 #define LCD_DATA2_PIN 4

pin for 4bit data bit 2

Definition at line 94 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.7 #define LCD_DATA2_PORT LCD_PORT

port for 4bit data bit 2

Definition at line 90 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.8 #define LCD_DATA3_PIN 5

pin for 4bit data bit 3

Definition at line 95 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.9 #define LCD_DATA3_PORT LCD_PORT

port for 4bit data bit 3

Definition at line 91 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.10 #define LCD_DISP_LENGTH 24

visibles characters per line of the display

Definition at line 62 of file lcd_display.h.

Referenced by lcd_putc().

5.7.3.11 #define LCD_E_PIN 6

pin for Enable line

Definition at line 101 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.12 #define LCD_E_PORT LCD_PORT

port for Enable line

Definition at line 100 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.13 #define LCD_IO_MODE 1

0: memory mapped mode, 1: IO port mode

Definition at line 71 of file lcd_display.h.

5.7.3.14 #define LCD_LINE_LENGTH 0x40

internal line length of the display

Definition at line 63 of file lcd_display.h.

5.7.3.15 #define LCD_LINES 2

number of visible lines of the display

Definition at line 61 of file lcd_display.h.

5.7.3.16 #define LCD_PORT PORTC

port for the LCD lines

Definition at line 87 of file lcd_display.h.

5.7.3.17 #define LCD_RS_PIN 7

pin for RS line

Definition at line 97 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.18 #define LCD_RS_PORT LCD_PORT

port for RS line

Definition at line 96 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.19 #define LCD_RW_PIN 3

pin for RW line

Definition at line 99 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.20 #define LCD_RW_PORT PORTD

port for RW line

Definition at line 98 of file lcd_display.h.

Referenced by lcd_init().

5.7.3.21 #define LCD_START_LINE1 0x00

DDRAM address of first char of line 1

Definition at line 64 of file lcd_display.h.

Referenced by lcd_gotoxy(), and lcd_putc().

5.7.3.22 #define LCD_START_LINE2 0x40

DDRAM address of first char of line 2

Definition at line 65 of file lcd_display.h.

Referenced by lcd_gotoxy(), and lcd_putc().

5.7.3.23 #define LCD_START_LINE3 0x14

DDRAM address of first char of line 3

Definition at line 66 of file lcd_display.h.

Referenced by lcd_gotoxy(), and lcd_putc().

5.7.3.24 #define LCD_START_LINE4 0x54

DDRAM address of first char of line 4

Definition at line 67 of file lcd_display.h.

Referenced by lcd_gotoxy(), and lcd_putc().

5.7.3.25 #define LCD_WRAP_LINES 0

0: no wrap, 1: wrap at end of visibile line

Definition at line 68 of file lcd_display.h.

5.7.3.26 #define XTAL 16000000

clock frequency in Hz, used to calculate delay timer

Definition at line 48 of file lcd_display.h.

5.7.4 Function Documentation

5.7.4.1 void lcd_clrscr (void)

Clear display and set cursor to home position.

Parameters:

void

Returns:

none

Definition at line 434 of file lcd_display.c.

References lcd_command().

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

5.7.4.2 void lcd_command (uint8_t *cmd*)

Send LCD controller instruction command.

Parameters:

cmd instruction to send to LCD controller, see HD44780 data sheet

Returns:

none

Definition at line 372 of file lcd_display.c.

Referenced by lcd_clrscr(), lcd_gotoxy(), lcd_home(), and lcd_init().

5.7.4.3 void lcd_data (uint8_t *data*)

Send data byte to LCD controller.

Similar to lcd_putc(), but without interpreting LF

Parameters:

data byte to send to LCD controller, see HD44780 data sheet

Returns:

none

Definition at line 384 of file lcd_display.c.

5.7.4.4 void lcd_gotoxy (uint8_t *x*, uint8_t *y*)

Set cursor to specified position.

Parameters:

x horizontal position
(0: left most position)

y vertical position
(0: first line)

Returns:

none

Definition at line 398 of file lcd_display.c.

References lcd_command(), LCD_START_LINE1, LCD_START_LINE2, LCD_START_LINE3, and LCD_START_LINE4.

Referenced by main().

5.7.4.5 void lcd_home (void)

Set cursor to home position.

Parameters:

void

Returns:

none

Definition at line 443 of file lcd_display.c.

References lcd_command().

5.7.4.6 void lcd_init (uint8_t *dispAttr*)

Initialize display and select type of cursor.

Parameters:

dispAttr **LCD_DISP_OFF** display off
LCD_DISP_ON display on, cursor off
LCD_DISP_ON_CURSOR display on, cursor on
LCD_DISP_ON_CURSOR_BLINK display on, cursor on flashing

Returns:

none

Definition at line 538 of file lcd_display.c.

References lcd_clrscr(), lcd_command(), LCD_DATA0_PIN, LCD_DATA0_PORT, LCD_DATA1_-PIN, LCD_DATA1_PORT, LCD_DATA2_PIN, LCD_DATA2_PORT, LCD_DATA3_PIN, LCD_DATA3_-PORT, LCD_E_PIN, LCD_E_PORT, LCD_RS_PIN, LCD_RS_PORT, LCD_RW_PIN, and LCD_RW_-PORT.

Referenced by init_all().

5.7.4.7 void lcd_putc (char *c*)

Display character at current cursor position.

Parameters:

c character to be displayed

Returns:

none

Definition at line 454 of file lcd_display.c.

References LCD_DISP_LENGTH, LCD_START_LINE1, LCD_START_LINE2, LCD_START_LINE3, and LCD_START_LINE4.

Referenced by lcd_puts(), and lcd_puts_p().

5.7.4.8 void lcd_puts (const char $*$ *s*)

Display string without auto linefeed.

Parameters:

s string to be displayed

Returns:

none

Definition at line 501 of file lcd_display.c.

References lcd_putc().

Referenced by init_all(), main(), and smartie_lcd_write_color().

5.7.4.9 void lcd_puts_p (const char $*$ *progmem_s*)

Display string from program memory without auto linefeed.

Parameters:

s string from program memory be be displayed

Returns:

none

See also:

lcd_puts_P

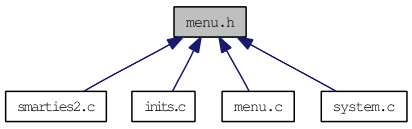
Definition at line 518 of file lcd_display.c.

References lcd_putc().

5.8 menu.h File Reference

The menu structure and handling.

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



Data Structures

- struct menu_entry_t

  The menu structure.

5.8.1 Detailed Description

The menu structure and handling.

Copyright (c) 2008 Simeon Felis

5.8.2 License

GPL2 Licence

5.8.3 Architecture

This is not an up to date, complete reference for the whole menu structure. However this is a rather completet explanation on how the menu works with examples.

The menu-control is locked to a 2-line alphanumeric display. It will be entered as soon as the smartie sorter is in SYS_MODE_PAUSE. Otherwise the lcd displays some status information:

During SYS_MODE_INIT :

```
+--------------+
| INITIALIZING |
|              |
+--------------+
```

During SYS_MODE_RUNNING :

```
+--------------+
| ENTER PAUSE  |
|RUN        COL|
+--------------+
```

During SYS_MODE_PAUSE : Greeting menu:

```
   right      +---------------+    right      +---------------+   right
 <---------> |  ENTER MENU   | <----------> |    RESUME     | <--------->
   left      |<prev   next >|    left      |<p          n>|   left
             +---------------+              +---------------+
               |                              |
               |enter_submenu()               |sys_resume()
```

MAIN menu:

```
  r    +---------------+    r    +---------------+    r    +---------------+    r    +---------------+    r
<---> | ROTATE REV    | <---> | ROTATE CATCH | <---> | TCS colors    | <---> |    Go Back    | <--->
  l    |<p           n>|    l    |<p           n>|    l    |<p           n>|    l    |<p           n>|
       +---------------+         +---------------+         +---------------+         +---------------+
       |                          |                          |                          |
       |sys_rotate_revolver()     |sys_rotate_catcher()      |sys_tcs_colors()          |enter_topmenu()
```

Each menu layer has its own array. Example:

MAIN menu

```
+---------------+  +---------------+  +---------------+ +---------------+
|      [0]      |  |      [1]      |  |      [2]      | |      [3]      |
|               |  |               |  |               | |               |
+---------------+  +---------------+  +---------------+ +---------------+
```

And so on.

The Display has two lines with 24 characters. Here the exact layout during SYS_MODE_RUNNING:

```
      1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
    + -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - +
  1 |                            T  I  T  L  E                                |
  2 | S  T  A  T  : [   M  O  D  E        ]       C  O  L  :[ C  O  L  O  R  ]|
    + -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - +
```

- Line 1: Column 1 to 24 is reserved for the title. If the push button is pressed, the action described by the title will be executed.
- Line 2:
  - Column 6 to 12 is reserved for the current mode. Following modes can be displayed:
    - PAUSE
    - RUNNING
  - Column 19 to 24 is reserved for the folling colors:
    - yellow: YELLOW
    - red: RED
    - blue: BLUE
    - brown: BROWN
    - green: GREEN
    - purple: PURPLE
    - unknown: UNKNOWN

Note: the column numbers may be out of date.

For some setting possibilties have a look at lcd_display.h

For lcd control functions have a look at lcd_display.c

The init of the menu is done at init_menu() in inits.c

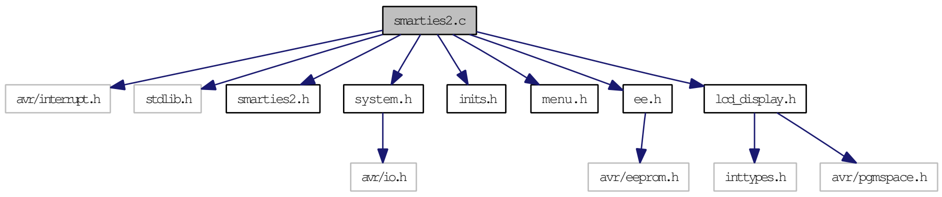
Definition in file menu.h.

5.9 smarties2.c File Reference

Entry file for Smarties project.

```
#include <avr/interrupt.h>
#include <stdlib.h>
#include "smarties2.h"
#include "system.h"
#include "inits.h"
#include "menu.h"
#include "ee.h"
#include "lcd_display.h"
```

Include dependency graph for smarties2.c:



Functions

- void smartie_lcd_write_color (smartie_color color)

  Writes the color to the current postion of the display.

- int main (void)

  Entry functin for smarties2.

5.9.1 Detailed Description

Entry file for Smarties project.

Copyright (c) 2008 Simeon Felis

Version:

0.1

Definition in file smarties2.c.

5.9.2 Function Documentation

5.9.2.1 int main (void)

Entry functin for smarties2.

This function controls the whole smarties machine by checking the current mode and performing the single steps required for the mode.

This function also handles the user inputs.

The subfunctions for driving the periphals are collected in system.h. The LCD menu is controlled by functions collected in lcd_display.h.

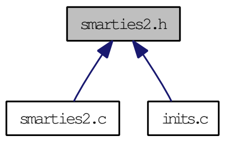
Definition at line 221 of file smarties2.c.

References catcher_rotate_absolute(), col_blue, col_brown, col_green, col_orange, col_pink, col_-purple, col_red, col_yellow, smartie_t::color, color_sensor_tcs_t::color, stepper_motor_t::current_pos, ee_memory_t::def_blu, ee_memory_t::def_gre, ee_memory_t::def_red, color_sensor_tcs_t::distance, color_sensor_tcs_t::filter_freq_blue, color_sensor_tcs_t::filter_freq_green, color_sensor_tcs_t::filter_-freq_none, color_sensor_tcs_t::filter_freq_red, system_step_t::I, system_step_t::II, system_step_t::III, init_all(), menu_entry_t::l_action, smartie_sorter_t::lb_revolver, lcd_clrscr(), lcd_gotoxy(), lcd_puts(), rotary_encoder_t::left, system_state_t::mode, system_state_t::mode_last, smartie_sorter_t::mot_catcher, smartie_sorter_t::mot_revolver, menu_entry_t::next, lightbarrier_t::passes, menu_entry_t::prev, smartie_-sorter_t::prog, prog_color_tcs, prog_none, prog_rotate_catcher, prog_rotate_revolver, prog_set_colors_-blue, prog_set_colors_brown, prog_set_colors_green, prog_set_colors_orange, prog_set_colors_pink, prog_set_colors_purple, prog_set_colors_red, prog_set_colors_restore, prog_set_colors_yellow, rotary_-encoder_t::push, menu_entry_t::r_action, REFERENCE_MEASURES, smartie_sorter_t::rev, revolver_-rotate_relative(), rotary_encoder_t::right, smartie_sorter_t::rotenc, smartie_sorter_t::sens_tcs, sensor_-tcs_get_color(), revolver_t::smart, smartie_lcd_write_color(), smartie_sorter_t::speed, stat_finished, stat_idle, stat_start_working, stat_stop_working, stat_working, smartie_sorter_t::state, vibrator_t::status, color_sensor_tcs_t::status, stepper_motor_t::status, vibrator_t::status_last, color_sensor_tcs_t::status_last, stepper_motor_t::status_last, system_state_t::step, sys_get_out_pos(), SYS_MODE_INIT, SYS_MODE_-PAUSE, SYS_MODE_RUNNING, stepper_motor_t::target_pos, menu_entry_t::text, ee_memory_t::usr_-blu, ee_memory_t::usr_gre, ee_memory_t::usr_red, smartie_sorter_t::vibr, and vibrator_start().

5.10 smarties2.h File Reference

Include file for smarties2 project.

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



Defines

- #define FALSE 0

  Useful boolean values FALSE.

- #define TRUE 1

  Useful boolean values TRUE.

- #define REFERENCE_MEASURES 5

  How many measures for a callibrating the reference value are required.

5.10.1 Detailed Description

Include file for smarties2 project.

Copyright (c) 2008 Simeon Felis

5.10.2 Description

This file includes all files necessary for the smarties2 project

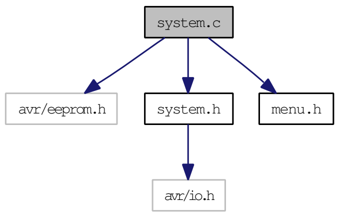
Definition in file smarties2.h.

5.11 system.c File Reference

Short routines for controlling the whole system and modules.

```
#include <avr/eeprom.h>
#include "system.h"
#include "menu.h"
```

Include dependency graph for system.c:



Functions

- void sys_enter_topmenu ()

  Will set the current menu to the upper menu layer.

- void sys_enter_submenu ()

  Will set the current menu to to lower menu layer.

- void sys_catcher_disable ()

  Diables (power off) the catcher stepper motor.

- void sys_catcher_enable ()

  Enables (power on) the catcher stepper motor.

- void sys_catcher_move_step ()

  Moves the catcher stepper motor for on step.

- void sys_catcher_rotate ()

  Rotates the catcher for one position.

- uint8_t sys_get_out_pos ()

  Gets the correct out index of the next to drop smartie.

- int8_t sys_catcher_is_lb_blocked ()

  Returns status for the catcher lightbarrier.

- void sys_revolver_rotate ()

  Rotates the revolver for one position.

- void sys_revolver_move_step ()

  Moves the revolver for one step.

- int8_t sys_revolver_is_lb_blocked ()

  Returns the status for the revolver lightbarrier.
- void sys_pause ()

  Will Enter the pause mode.
- void sys_reference_measure_blue ()

  Initiates color callibration for corresponding smartie color.
- void sys_reference_measure_green ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_red ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_yellow ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_orange ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_brown ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_purple ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_pink ()

  See sys_reference_measure_blue ().
- void sys_reference_measure_restore ()

  Initiates to restore the color reference values.
- void sys_resume ()

  Leave the pause mode.
- void sys_set_speed ()

  Set the operating speed of the sorter to the current demand value.
- void sys_speed_up ()

  Set the operating speed of the sorter up by decreasing pause times.
- void sys_speed_down ()

  Set the operating speed of the sorter down by increasing pause times.
- void sys_measure_tcs ()

  Initiate a color measurement with the tcs color sensor.
- void sys_measure_adjd ()

  Initiate a color measurement with the adjd color sensor.

- void vibrator_start ()

  Initiates the vibrator to start.

- void sensor_adjd_get_color ()

  Initiates the color detection by the ADJD color sensor.

- void sensor_tcs_get_color ()

  Initiates the color detection by the TCS color sensor.

- void catcher_rotate_absolute (smartie_color move_to)

  Rotates the catcher to a certain position.

- void catcher_rotate_relative (int8_t rel_pos)

  Rotates the catcher from the current position to a relative next position regarding 'hole above hole'.

- void revolver_rotate_relative (int8_t rel_pos)

  Rotates the revolver from the current position to a relative next position regarding 'hole above hole'.

5.11.1 Detailed Description

Short routines for controlling the whole system and modules.

Definition in file system.c.

5.11.2 Function Documentation

5.11.2.1 void catcher_rotate_absolute (smartie_color *move_to*)

Rotates the catcher to a certain position.

This function will rotate the catcher to position specified by the a color smartie_color_t.

Definition at line 346 of file system.c.

References CATCH_MAX_SIZE, catcher_rotate_relative(), col_unknown, stepper_motor_t::current_pos, smartie_sorter_t::mot_catcher, stat_finished, and stepper_motor_t::status_last.

Referenced by main().

5.11.2.2 void catcher_rotate_relative (int8_t *rel_pos*)

Rotates the catcher from the current position to a relative next position regarding 'hole above hole'.

Most of the work is being done in motor_stuff.

Warning: This function does not check if the catcher is already working. Before calling this function, check if the motor's status and last status equals stat_idle. It also doesn't check the parameter, if the value is reasonable.

Parameters:

rel_pos The position where to move to. The value of rel_pos will be not checked, so the value must be lower than REV_MAX_SIZE or CATCH_MAX_SIZE.

Definition at line 384 of file system.c.

References CATCH_MAX_SIZE, stepper_motor_t::current_pos, smartie_sorter_t::mot_catcher, stat_-finished, stat_start_working, stepper_motor_t::status, stepper_motor_t::status_last, and stepper_motor_-t::target_pos.

Referenced by catcher_rotate_absolute(), init_motors(), and sys_catcher_rotate().

5.11.2.3 void revolver_rotate_relative (int8_t *rel_pos*)

Rotates the revolver from the current position to a relative next position regarding 'hole above hole'.

Most of the work is being done in motor_stuff.

Warning: This function does not check if the revolver is already working. Before calling this function, check if the motor's status and last status equals stat_idle. It also doesn't check the parameter, if the value is reasonable.

Parameters:

rel_pos The position where to move to. The value of rel_pos will be not checked, so the value must be lower than REV_MAX_SIZE or CATCH_MAX_SIZE.

Definition at line 415 of file system.c.

References stepper_motor_t::current_pos, smartie_sorter_t::mot_revolver, REV_MAX_SIZE, stat_-finished, stat_start_working, stepper_motor_t::status, stepper_motor_t::status_last, and stepper_motor_-t::target_pos.

Referenced by init_motors(), main(), and sys_revolver_rotate().

5.11.2.4 void sys_catcher_disable ()

Diables (power off) the catcher stepper motor.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 44 of file system.c.

References CATCH_DISABLE.

Referenced by init_motors().

5.11.2.5 void sys_catcher_enable ()

Enables (power on) the catcher stepper motor.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 54 of file system.c.

References CATCH_ENABLE.

Referenced by init_motors().

5.11.2.6 int8_t sys_catcher_is_lb_blocked ()

Returns status for the catcher lightbarrier.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 98 of file system.c.

References IS_LB_CATCHER.

Referenced by init_motors().

5.11.2.7 void sys_catcher_rotate ()

Rotates the catcher for one position.

This function is intened for manual usage and should not be used during normal running mode SYS_-MODE_RUNNING, only during SYS_MODE_PAUSE.

Definition at line 72 of file system.c.

References catcher_rotate_relative(), smartie_sorter_t::mot_catcher, smartie_sorter_t::prog, prog_rotate_-catcher, stat_idle, stepper_motor_t::status, and stepper_motor_t::status_last.

Referenced by init_menu().

5.11.2.8 void sys_enter_submenu ()

Will set the current menu to to lower menu layer.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 33 of file system.c.

References smartie_sorter_t::prog, prog_none, and menu_entry_t::submenu.

Referenced by init_menu().

5.11.2.9 void sys_enter_topmenu ()

Will set the current menu to the upper menu layer.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 22 of file system.c.

References smartie_sorter_t::prog, prog_none, and menu_entry_t::topmenu.

Referenced by init_menu().

5.11.2.10 uint8_t sys_get_out_pos ()

Gets the correct out index of the next to drop smartie.

Returns:

The index of smartie in the revolver which will be dropped next

Definition at line 85 of file system.c.

References stepper_motor_t::current_pos, smartie_sorter_t::mot_revolver, and REV_MAX_SIZE.

Referenced by main().

5.11.2.11 void sys_pause ()

Will Enter the pause mode.

This function will enter the pause mode after the current function is finished. This function is usually called from the lcd menu. This function is only available in the mode SYS_MODE_RUNNING

Definition at line 149 of file system.c.

References system_state_t::mode, system_state_t::mode_last, smartie_sorter_t::state, and SYS_MODE_-PAUSE.

Referenced by init_menu().

5.11.2.12 void sys_reference_measure_blue ()

Initiates color callibration for corresponding smartie color.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 163 of file system.c.

References smartie_sorter_t::prog, and prog_set_colors_blue.

Referenced by init_menu().

5.11.2.13 void sys_reference_measure_restore ()

Initiates to restore the color reference values.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 222 of file system.c.

References smartie_sorter_t::prog, and prog_set_colors_restore.

Referenced by init_menu().

5.11.2.14 void sys_resume ()

Leave the pause mode.

This function will leave the pause mode. This function is usually called from the lcd menu. This function is only available in the mode SYS_MODE_PAUSE

Definition at line 233 of file system.c.

References system_state_t::mode, system_state_t::mode_last, smartie_sorter_t::state, and SYS_MODE_-RUNNING.

Referenced by init_menu().

5.11.2.15 int8_t sys_revolver_is_lb_blocked ()

Returns the status for the revolver lightbarrier.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 138 of file system.c.

References IS_LB_REVOLVER.

Referenced by init_motors().

5.11.2.16 void sys_revolver_rotate ()

Rotates the revolver for one position.

This function is intened for manual usage and should not be used during normal running mode SYS_-MODE_RUNNING, only during SYS_MODE_PAUSE.

Definition at line 118 of file system.c.

References smartie_sorter_t::mot_revolver, smartie_sorter_t::prog, prog_rotate_revolver, revolver_rotate_-relative(), stat_idle, stepper_motor_t::status, and stepper_motor_t::status_last.

Referenced by init_menu().

5.11.2.17 void sys_speed_down ()

Set the operating speed of the sorter down by increasing pause times.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 269 of file system.c.

References smartie_sorter_t::prog, smartie_sorter_t::speed, and sys_set_speed().

Referenced by init_menu().

5.11.2.18 void sys_speed_up ()

Set the operating speed of the sorter up by decreasing pause times.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 256 of file system.c.

References smartie_sorter_t::prog, smartie_sorter_t::speed, and sys_set_speed().

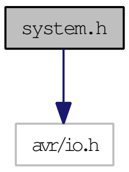
Referenced by init_menu().

5.12 system.h File Reference

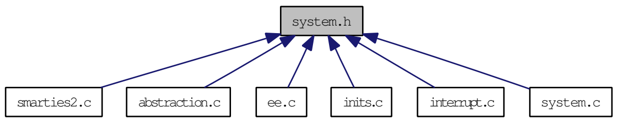
Smarites machine API header file.

`#include <avr/io.h>`

Include dependency graph for system.h:



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



Data Structures

- struct system_step_t

  The single steps during running mode SYS_MODE_RUNNING.

- struct system_state_t

  Grouping of mode and steps.

- struct rotary_encoder_t

  The rotary encoder (user input device) structure.

- struct smartie_t

  Describes the properties a smartie, which is transported by the revolver_t, can have.

- struct lightbarrier_t

  Describes the module lightbarrier.

- struct stepper_motor_t

  The stepper motor.

- struct color_sensor_adjd_t

  Describes the ADJD-S371 color sensor.

- struct color_sensor_tcs_t

  Describes the TCS230 color sensor.

- struct vibrator_t

  Describes the virbrator (shaker) module.

- struct revolver_t

    Describes the revolver module (the disc).

- struct smartie_sorter_t

    All devices from the smartie sorter collected to one bundle.

Defines

- #define COL_SENS_ADJD_LED_PORT PORTA

    The ADJD color sensor Port for the LED.

- #define COL_SENS_ADJD_LED_BIT PA0

    The ADJD color sensor Portbit for the LED.

- #define COL_SENS_ADJD_LED_ON (COL_SENS_ADJD_LED_PORT |= (1<<COL_SENS_-ADJD_LED_BIT))

    Switches the LED of the ADJD color sensor on.

- #define COL_SENS_ADJD_LED_OFF (COL_SENS_ADJD_LED_PORT &= ~(1<<COL_-SENS_ADJD_LED_BIT))

    Switches the LED of the ADJD color sensor off.

- #define COL_SENS_TCS_IN_PORT PORTD

    Input port for TCS color sensor.

- #define COL_SENS_TCS_IN_ICP PD2

    ICP pin for TCS color sensor (clock signal).

- #define COL_SENS_TCS_OUT_PORT PORTA

    Output port for TCS color sensor.

- #define COL_SENS_TCS_S2_BIT PA7

    S2 settings pin for TCS color sensor.

- #define COL_SENS_TCS_S0_BIT PA6

    S0 settings pin for TCS color sensor.

- #define COL_SENS_TCS_S3_BIT PA5

    S3 settings pin for TCS color sensor.

- #define COL_SENS_TCS_S1_BIT PA4

    S1 settings pin for TCS color sensor.

- #define COL_SENS_TCS_OE_BIT PA3

    Output Portbit enable for TCS color sensor.

- #define COL_SENS_TCS_SAMPLE_TIME 30

Time to measure TCS OUT frequency in milliseconds.

- #define COL_SENS_TCS_ENABLE (COL_SENS_TCS_OUT_PORT &= ∼(1<<COL_SENS_-TCS_OE_BIT))

  Enables the TCS color sensor output clk.

- #define COL_SENS_TCS_DISABLE (COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_-TCS_OE_BIT))

  Disables the TCS color sensor output clk.

- #define COL_SENS_TCS_FREQ_MESURE_EN (GICR |= (1<<INT0))

  Enables interrupt for counting slopes (falling) from the TCS OUT pin.

- #define COL_SENS_TCS_FREQ_MESURE_DI (GICR &= ∼(1<<INT0))

  Enables interrupt for counting slopes (falling) from the TCS OUT pin.

- #define COL_SENS_TCS_SET_FREQ_SCALE(percentage)

  Sets the frequency scaler for the TCS color sensor.

- #define COL_SENS_TCS_SET_FILTER(color)

  Sets the color filter for the TCS color sensor.

- #define STEPPER_PORT PORTD

  Output port for stepper motors.

- #define REV_BIT_EN PD7

  Portbit for Enable driver for revolver stepper motor.

- #define REV_BIT_CW PD6

  Portbit for Rotate direction for revolver stepper motor (CW/CCW).

- #define REV_BIT_CLK PD5

  Portbit for Clock signal for driver for revolver stepper motor.

- #define REV_MOVE_STEP

  Move the revolver stepper motor for one single step.

- #define REV_ENABLE (STEPPER_PORT |= (1<<REV_BIT_EN))

  Enables the driver for the stepper motor.

- #define REV_DISABLE (STEPPER_PORT &= ∼(1<<REV_BIT_EN))

  Disables the driver for the stepper motor.

- #define REV_SET_CW (STEPPER_PORT &= ∼(1<<REV_BIT_CW))

  Rotating direction clockwise.

- #define REV_SET_CCW (STEPPER_PORT |= (1<<REV_BIT_CW))

  Rotating directino conter clock wise.

- #define REV_STEP_DURATION 20

Duration of one step in milliseconds. This value controlles the rotating speed.

- #define REV_RAMP_DURATION 1

  Duration of the ramp up or ramp down in steps.

- #define REV_MAX_SIZE 12

  The amount of smarties (holes) which fit into the revolver.

- #define REV_STEPS_ESTIMATED 5

  Amount of steps for each positions 'hole abouve hole'.

- #define CATCH_BIT_EN PD4

  Portbit for Enable driver for catcher stepper motor.

- #define CATCH_BIT_CW PD1

  Portbit for Rotate direction for catcher stepper motor (CW/CC.

- #define CATCH_BIT_CLK PD0

  Portbit for Clock signal for driver for catcher stepper motorW).

- #define CATCH_MOVE_STEP

  Move the catcher stepper motor for one single step.

- #define CATCH_ENABLE (STEPPER_PORT |= (1<<CATCH_BIT_EN))

  Enables the driver for the steper motor.

- #define CATCH_DISABLE (STEPPER_PORT &= ~(1<<CATCH_BIT_EN))

  Disables the driver for the steper motor.

- #define CATCH_SET_CW (STEPPER_PORT &= ~(1<<CATCH_BIT_CW))

  Rotating direction clockwise.

- #define CATCH_SET_CCW (STEPPER_PORT |= (1<<CATCH_BIT_CW))

  Rotating directino conter clock wise.

- #define CATCH_STEP_DURATION 10

  Duration of one step in one Millisecond. This value controlles the rotating speed.

- #define CATCH_RAMP_DURATION 2

  Duration of the ramp up or ramp down in steps.

- #define CATCH_MAX_SIZE 9

  The amount of catcher tubes for sorting the smarties.

- #define CATCH_STEPS_ESTIMATED 14

  Amount of steps for each positions 'hole abouve hole'.

- #define VIBR_PORT PORTB

  The Vibrator Port.

- #define VIBR_BIT PB3

  The Vibrator Portbit.

- #define VIBR_ON (VIBR_PORT |= (1<<VIBR_BIT))

  Switches the Vibrator on.

- #define VIBR_OFF (VIBR_PORT &=~(1<<VIBR_BIT))

  Switches the Bibrator off.

- #define VIBR_DURATION (700)

  Default duration for shaker (vibrator).

- #define ROTENC_PORT PORTB

  Rotary encoder port (output) for AB signals.

- #define ROTENC_A_BIT PB0

  Rotary encoder signal A (Pin 5 of connector).

- #define ROTENC_B_BIT PB1

  Rotary encoder signal B (Pin 4 of connector).

- #define ROTENC_PUSH_BIT PB2

  Rotary encoder pushbutton signal (Pin 2 of connector).

- #define ROTENC_INIT() (ROTENC_DDR &= ~((1<<ROTENC_A_BIT) | (1<<ROTENC_B_-BIT) | (1<<ROTENC_PUSH_BIT)))

  Initializes input periphals for Rotary encoder.

- #define IS_ROTENC_A ((ROTENC_PIN & (1<<ROTENC_A_BIT)))

  Output status of rotary encoder.

- #define IS_ROTENC_B ((ROTENC_PIN & (1<<ROTENC_B_BIT)))

  Output status of rotary encoder.

- #define IS_ROTENC_AB (IS_ROTENC_A && IS_ROTENC_B)

  Output status of rotary encoder.

- #define IS_ROTENC_NONE (!IS_ROTENC_A && !IS_ROTENC_B)

  Output status of rotary encoder.

- #define IS_ROTENC_PUSH (ROTENC_PIN & (1<<ROTENC_PUSH_BIT))

  Output status of rotary encoder.

- #define LB_PORT PORTA

  Lightbarriere Port.

- #define LB_BIT_CATCH PA2

  Lightbarriere Catcher Positinoer Portbit.

- #define LB_BIT_REV PA1

Lightbarriere Revolver Positioner Portbit.

- #define IS_LB_CATCHER (!(LB_PIN & (1<<LB_BIT_CATCH)))

  Returns TRUE if lightbarrier is blocked.

- #define IS_LB_REVOLVER (!(LB_PIN & (1<<LB_BIT_REV)))

  Returns TRUE if lightbarrier is blocked.

Typedefs

- typedef float color_avarage [col_unknown]

  Stores the color reference values of one channel for all smarties.

Enumerations

- enum system_mode_t { SYS_MODE_INIT = 0, SYS_MODE_PAUSE, SYS_MODE_RUNNING }

  The mode of the machine.

- enum common_stat_t {
  stat_idle = 0, stat_start_working, stat_working, stat_stop_working,
  stat_finished }

  The status a device can have.

- enum program_t {
  prog_none = 0, prog_rotate_catcher, prog_rotate_revolver, prog_color_tcs ,
  prog_set_colors_blue, prog_set_colors_green, prog_set_colors_red, prog_set_colors_yellow,
  prog_set_colors_orange, prog_set_colors_brown, prog_set_colors_purple, prog_set_colors_pink,
  prog_set_colors_restore }

  Programs which are executed during SYS_MODE_PAUSE.

- enum rotary_encoder_status_t {
  ROTENC_NONE = 0, ROTENC_PUSH, ROTENC_A, ROTENC_B,
  ROTENC_BOTH }

  The rotary encoder's elements can have following status.

- enum smartie_color_t {
  col_blue = 0, col_green, col_red, col_yellow,
  col_orange, col_brown, col_purple, col_pink,
  col_unknown }

  All supported colors.

- enum lightbarrier_status_t { lb_free = 0, lb_blocked }

  The status a lightbarrier can have.

Functions

- void sys_enter_topmenu ()

    Will set the current menu to the upper menu layer.

- void sys_enter_submenu ()

    Will set the current menu to to lower menu layer.

- void sys_catcher_enable ()

    Enables (power on) the catcher stepper motor.

- void sys_catcher_disable ()

    Diables (power off) the catcher stepper motor.

- void sys_catcher_move_step ()

    Moves the catcher stepper motor for on step.

- void sys_catcher_rotate ()

    Rotates the catcher for one position.

- int8_t sys_catcher_is_lb_blocked ()

    Returns status for the catcher lightbarrier.

- uint8_t sys_get_out_pos ()

    Gets the correct out index of the next to drop smartie.

- void sys_revolver_move_step ()

    Moves the revolver for one step.

- void sys_revolver_rotate ()

    Rotates the revolver for one position.

- int8_t sys_revolver_is_lb_blocked ()

    Returns the status for the revolver lightbarrier.

- void sys_pause ()

    Will Enter the pause mode.

- void sys_resume ()

    Leave the pause mode.

- void sys_reference_measure_blue ()

    Initiates color callibration for corresponding smartie color.

- void sys_reference_measure_green ()

    See sys_reference_measure_blue ().

- void sys_reference_measure_red ()

    See sys_reference_measure_blue ().

- void sys_reference_measure_yellow ()

  See sys_reference_measure_blue ().

- void sys_reference_measure_orange ()

  See sys_reference_measure_blue ().

- void sys_reference_measure_brown ()

  See sys_reference_measure_blue ().

- void sys_reference_measure_pink ()

  See sys_reference_measure_blue ().

- void sys_reference_measure_purple ()

  See sys_reference_measure_blue ().

- void sys_reference_measure_restore ()

  Initiates to restore the color reference values.

- void sys_set_speed ()

  Set the operating speed of the sorter to the current demand value.

- void sys_speed_up ()

  Set the operating speed of the sorter up by decreasing pause times.

- void sys_speed_down ()

  Set the operating speed of the sorter down by increasing pause times.

- void sys_measure_tcs ()

  Initiate a color measurement with the tcs color sensor.

- void sys_measure_adjd ()

  Initiate a color measurement with the adjd color sensor.

- void sensor_adjd_get_color ()

  Initiates the color detection by the ADJD color sensor.

- void sensor_tcs_get_color ()

  Initiates the color detection by the TCS color sensor.

- void catcher_rotate_absolute (smartie_color move_to)

  Rotates the catcher to a certain position.

- void catcher_rotate_relative (int8_t)

  Rotates the catcher from the current position to a relative next position regarding 'hole above hole'.

- void revolver_rotate_relative (int8_t rel_pos)

  Rotates the revolver from the current position to a relative next position regarding 'hole above hole'.

- void vibrator_start ()

  Initiates the vibrator to start.

5.12.1 Detailed Description

Smarites machine API header file.

Copyright (c) 2008 Simeon Felis

5.12.2 License

GPL2 Licence

Here are many IO related functions declared

Definition in file system.h.

5.12.3 Define Documentation

5.12.3.1 #define COL_SENS_TCS_SET_FILTER(color)

Value:

```
do {                                                        \
        if (color == col_red) {                                 \
            COL_SENS_TCS_OUT_PORT &= ~(1<<COL_SENS_TCS_S2_BIT); \
            COL_SENS_TCS_OUT_PORT &= ~(1<<COL_SENS_TCS_S3_BIT); \
        }                                                       \
        else if (color == col_blue) {                           \
            COL_SENS_TCS_OUT_PORT &= ~(1<<COL_SENS_TCS_S2_BIT); \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S3_BIT);  \
        }                                                       \
        else if (color == col_green) {                          \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S2_BIT);  \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S3_BIT);  \
        }                                                       \
        else if (color == col_unknown) {                        \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S2_BIT);  \
            COL_SENS_TCS_OUT_PORT &= ~(1<<COL_SENS_TCS_S3_BIT); \
        }                                                       \
    } while (0)
```

Sets the color filter for the TCS color sensor.

Parameters:

color	**col_red** Red filter
	col_blue Blue filter
	col_green Green filter on
	col_unknown No filter

Definition at line 116 of file system.h.

Referenced by sensor_tcs_stuff().

5.12.3.2 #define COL_SENS_TCS_SET_FREQ_SCALE(percentage)

Value:

```
do {                                                        \
        COL_SENS_TCS_OUT_PORT &= ~(1<<COL_SENS_TCS_S1_BIT);     \
        COL_SENS_TCS_OUT_PORT &= ~(1<<COL_SENS_TCS_S0_BIT);     \
        if (percentage == 2) {                                  \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S1_BIT);  \
        }                                                       \
        else if (percentage == 20) {                            \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S0_BIT);  \
        }                                                       \
        else if (percentage == 100) {                           \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S0_BIT);  \
            COL_SENS_TCS_OUT_PORT |= (1<<COL_SENS_TCS_S1_BIT);  \
        }                                                       \
    } while (0)
```

Sets the frequency scaler for the TCS color sensor.

Parameters:

percentage **2** 2%
20 20%
100 100%

Definition at line 92 of file system.h.

Referenced by init_sensor_tcs().

5.12.4 Enumeration Type Documentation

5.12.4.1 enum common_stat_t

The status a device can have.

The different modules of the smartie sorter can have different operating status. You can recognize if a module has just finished by checking the last status.

Enumerator:

stat_idle The module/device is doing nothing and is ready for a new job.
stat_start_working Setting a module's state to this will start the device to work.
stat_working Indicates the device is busy. The device will automatically change to this state.
stat_stop_working Setting a device to this state will stop the device.
stat_finished Indicates the device is finished. The device will automatically change to this state.

Definition at line 252 of file system.h.

5.12.4.2 enum lightbarrier_status_t

The status a lightbarrier can have.

Enumerator:

lb_free Nothing inbetween the lightbarrier.
lb_blocked The lightbarrier is blocked.

Definition at line 354 of file system.h.

5.12.4.3 enum program_t

Programs which are executed during SYS_MODE_PAUSE.

Enumerator:

prog_none No program executed.

prog_rotate_catcher Indicates that the catcher rotate program is running.

prog_rotate_revolver Indicates that the revolver rotate program is running.

prog_color_tcs Indicates that the color measure program is runnging.

prog_set_colors_blue Calibrates the color refence values. Halts the state machine.

prog_set_colors_green Calibrates the color refence values. Halts the state machine.

prog_set_colors_red Calibrates the color refence values. Halts the state machine.

prog_set_colors_yellow Calibrates the color refence values. Halts the state machine.

prog_set_colors_orange Calibrates the color refence values. Halts the state machine.

prog_set_colors_brown Calibrates the color refence values. Halts the state machine.

prog_set_colors_purple Calibrates the color refence values. Halts the state machine.

prog_set_colors_pink Calibrates the color refence values. Halts the state machine.

prog_set_colors_restore Resores the color refernce values to system defaults. Halts the state machine.

Definition at line 281 of file system.h.

5.12.4.4 enum rotary_encoder_status_t

The rotary encoder's elements can have following status.

Enumerator:

ROTENC_NONE Applicable for Push button, A-Pin and B-Pin.

ROTENC_PUSH The Button is pushed.

ROTENC_A Currently only the A-Pin of the rotary encoder is set.

ROTENC_B Currently only the B-Pin of the rotary encoder is set.

ROTENC_BOTH Currently both, the A-Pin and B-Pin of the rotary encoder are set.

Definition at line 302 of file system.h.

5.12.4.5 enum smartie_color_t

All supported colors.

This enum is used for indexing the reference color tables, the positioning of the catcher and may be more. col_unknown is often used as end conditions for loops. In the color tables, this enums elements represent the index for color rows.

Enumerator:

col_blue Blue.

col_green Green.

col_red Red.

col_yellow Yellow.

col_orange Orange.

col_brown Brown.

col_purple Purple.

col_pink Pink.

col_unknown Indexed as last color (highest counter). Insert colors above this one!

Definition at line 329 of file system.h.

5.12.4.6 enum system_mode_t

The mode of the machine.

Enumerator:

SYS_MODE_INIT After reset or power on.

SYS_MODE_PAUSE Pausing the smartie sorter and operated manually.

SYS_MODE_RUNNING Smartie sorter is running automatically.

Definition at line 239 of file system.h.

5.12.5 Function Documentation

5.12.5.1 void catcher_rotate_absolute (smartie_color *move_to*)

Rotates the catcher to a certain position.

This function will rotate the catcher to position specified by the a color smartie_color_t.

Definition at line 346 of file system.c.

References CATCH_MAX_SIZE, catcher_rotate_relative(), col_unknown, stepper_motor_t::current_pos, smartie_sorter_t::mot_catcher, stat_finished, and stepper_motor_t::status_last.

Referenced by main().

5.12.5.2 void catcher_rotate_relative (int8_t *rel_pos*)

Rotates the catcher from the current position to a relative next position regarding 'hole above hole'.

Most of the work is being done in motor_stuff.

Warning: This function does not check if the catcher is already working. Before calling this function, check if the motor's status and last status equals stat_idle. It also doesn't check the parameter, if the value is reasonable.

Parameters:

rel_pos The position where to move to. The value of rel_pos will be not checked, so the value must be lower than REV_MAX_SIZE or CATCH_MAX_SIZE.

Definition at line 384 of file system.c.

References CATCH_MAX_SIZE, stepper_motor_t::current_pos, smartie_sorter_t::mot_catcher, stat_-finished, stat_start_working, stepper_motor_t::status, stepper_motor_t::status_last, and stepper_motor_-t::target_pos.

Referenced by catcher_rotate_absolute(), init_motors(), and sys_catcher_rotate().

5.12.5.3 void revolver_rotate_relative (int8_t *rel_pos*)

Rotates the revolver from the current position to a relative next position regarding 'hole above hole'.

Most of the work is being done in motor_stuff.

Warning: This function does not check if the revolver is already working. Before calling this function, check if the motor's status and last status equals stat_idle. It also doesn't check the parameter, if the value is reasonable.

Parameters:

rel_pos The position where to move to. The value of rel_pos will be not checked, so the value must be lower than REV_MAX_SIZE or CATCH_MAX_SIZE.

Definition at line 415 of file system.c.

References stepper_motor_t::current_pos, smartie_sorter_t::mot_revolver, REV_MAX_SIZE, stat_-finished, stat_start_working, stepper_motor_t::status, stepper_motor_t::status_last, and stepper_motor_-t::target_pos.

Referenced by init_motors(), main(), and sys_revolver_rotate().

5.12.5.4 void sys_catcher_disable ()

Diables (power off) the catcher stepper motor.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 44 of file system.c.

References CATCH_DISABLE.

Referenced by init_motors().

5.12.5.5 void sys_catcher_enable ()

Enables (power on) the catcher stepper motor.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 54 of file system.c.

References CATCH_ENABLE.

Referenced by init_motors().

5.12.5.6 int8_t sys_catcher_is_lb_blocked ()

Returns status for the catcher lightbarrier.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 98 of file system.c.

References IS_LB_CATCHER.

Referenced by init_motors().

5.12.5.7 void sys_catcher_rotate ()

Rotates the catcher for one position.

This function is intened for manual usage and should not be used during normal running mode SYS_-MODE_RUNNING, only during SYS_MODE_PAUSE.

Definition at line 72 of file system.c.

References catcher_rotate_relative(), smartie_sorter_t::mot_catcher, smartie_sorter_t::prog, prog_rotate_-catcher, stat_idle, stepper_motor_t::status, and stepper_motor_t::status_last.

Referenced by init_menu().

5.12.5.8 void sys_enter_submenu ()

Will set the current menu to to lower menu layer.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 33 of file system.c.

References smartie_sorter_t::prog, prog_none, and menu_entry_t::submenu.

Referenced by init_menu().

5.12.5.9 void sys_enter_topmenu ()

Will set the current menu to the upper menu layer.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 22 of file system.c.

References smartie_sorter_t::prog, prog_none, and menu_entry_t::topmenu.

Referenced by init_menu().

5.12.5.10 uint8_t sys_get_out_pos ()

Gets the correct out index of the next to drop smartie.

Returns:

The index of smartie in the revolver which will be dropped next

Definition at line 85 of file system.c.

References stepper_motor_t::current_pos, smartie_sorter_t::mot_revolver, and REV_MAX_SIZE.

Referenced by main().

5.12.5.11 void sys_pause ()

Will Enter the pause mode.

This function will enter the pause mode after the current function is finished. This function is usually called from the lcd menu. This function is only available in the mode SYS_MODE_RUNNING

Definition at line 149 of file system.c.

References system_state_t::mode, system_state_t::mode_last, smartie_sorter_t::state, and SYS_MODE_-PAUSE.

Referenced by init_menu().

5.12.5.12 void sys_reference_measure_blue ()

Initiates color callibration for corresponding smartie color.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 163 of file system.c.

References smartie_sorter_t::prog, and prog_set_colors_blue.

Referenced by init_menu().

5.12.5.13 void sys_reference_measure_restore ()

Initiates to restore the color reference values.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 222 of file system.c.

References smartie_sorter_t::prog, and prog_set_colors_restore.

Referenced by init_menu().

5.12.5.14 void sys_resume ()

Leave the pause mode.

This function will leave the pause mode. This function is usually called from the lcd menu. This function is only available in the mode SYS_MODE_PAUSE

Definition at line 233 of file system.c.

References system_state_t::mode, system_state_t::mode_last, smartie_sorter_t::state, and SYS_MODE_-RUNNING.

Referenced by init_menu().

5.12.5.15 int8_t sys_revolver_is_lb_blocked ()

Returns the status for the revolver lightbarrier.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 138 of file system.c.

References IS_LB_REVOLVER.

Referenced by init_motors().

5.12.5.16 void sys_revolver_rotate ()

Rotates the revolver for one position.

This function is intened for manual usage and should not be used during normal running mode SYS_-MODE_RUNNING, only during SYS_MODE_PAUSE.

Definition at line 118 of file system.c.

References smartie_sorter_t::mot_revolver, smartie_sorter_t::prog, prog_rotate_revolver, revolver_rotate_-relative(), stat_idle, stepper_motor_t::status, and stepper_motor_t::status_last.

Referenced by init_menu().

5.12.5.17 void sys_speed_down ()

Set the operating speed of the sorter down by increasing pause times.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 269 of file system.c.

References smartie_sorter_t::prog, smartie_sorter_t::speed, and sys_set_speed().

Referenced by init_menu().

5.12.5.18 void sys_speed_up ()

Set the operating speed of the sorter up by decreasing pause times.

This function is prepared especially for functions started from the menu. This function works only if the menu is correct initialized.

Definition at line 256 of file system.c.

References smartie_sorter_t::prog, smartie_sorter_t::speed, and sys_set_speed().

Referenced by init_menu().

Index