e-puck Reference Manual

1.0

Generated by Doxygen 1.5.4

Fri Nov 9 06:32:57 2007

Contents

Chapter 1

e-puck standard library documentation

1.1 Introduction

This project has been started at the Ecole Polytechnique Federale de Lausanne as collaboration between the Autonomous Systems Lab, the Swarm-Intelligent Systems group and the Laboratory of Intelligent System.

An educational robot: The main goal of this project is to develop a miniature mobile robot for educational purposes at university level. To achieve this goal the robot needs, in our opinion, the following features:

- Good structure. The robot should have a clean mechanical structure, simple to understand. The electronics, processor structure and software have to be a good example of a clean modern system.
- Flexibility. The robot should cover a large spectrum of educational activities and should therefore have a large potential in its sensors, processing power and extensions. Potential educational fields are, for instance, mobile robotics, real-time programming, embedded systems, signal processing, image or sound feature extraction, human-machine interaction or collective systems.
- User friendly. The robot should be small and easy to exploit on a table next to a computer. It should need minimal wiring, battery operation and optimal working comfort.
- Good robustness and simple maintenance. The robot should resist to student use and be simple and cheap to repair.
- Cheap. The robot, for large use, should be cheap (450-550 euros)

1.2 Documentation organization

This documentation is divided in five sections (as you can see on the top of the page):

- Main Page: The startup page.
- Modules: An overview of all the modules that compose this library. Here you can see all the files containing by each module and a detailed description of each module. Look at these pages to have a better idea of what each module is doing.
- Data Structures: Here are listed all the C-struct of the library.
- Files: All the library's files listed by alphabetical order.
- Directories: The directories architectures of the library.

1.3 External links

- http://www.e-puck.org/ The official site of the e-puck
- https://gna.org/projects/e-puck/ The developers area at gna
- http://lsro.epfl.ch/ The site of the lab where the e-puck has been created
- http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45 The license

Chapter 2

e-puck Module Index

2.1 e-puck Modules

Here is a list of all modules:

Chapter 3

e-puck Directory Hierarchy

3.1 e-puck Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 4

e-puck Data Structure Index

4.1 e-puck Data Structures

Here are the data structures with brief descriptions:

Chapter 5

e-puck File Index

5.1 e-puck File List

Here is a list of all files with brief descriptions:

Chapter 6

e-puck Module Documentation

6.1 Analogic/Digital conversion (ADC)

Files

- file **e_acc.c**

  Accessing the accelerometer data.

- file **e_acc.h**

  Accessing the accelerometer data.

- file **e_ad_conv.c**

  Module for the advance Analogic/Digital conversion.

- file **e_ad_conv.h**

  Module for the advance Analogic/Digital conversion.

- file **e_micro.c**

  Accessing the microphone data.

- file **e_micro.h**

  Accessing the microphone data.

- file **e_prox.c**

  Accessing proximity sensor of e-puck.

- file **e_prox.h**

  Accessing proximity sensor of e-puck.

- file **e_accelerometer.c**

  Accessing the accelerometer sensor data.

- file **e_accelerometer.h**

  Accessing the accelerometer sensor data.

- file **e_ad_conv.c**

  Module for the Analogic/Digital conversion.

- file **e_ad_conv.h**

  Module for the Analogic/Digital conversion.

- file **e_micro.c**

  Accessing the microphone data.

- file **e_micro.h**

  Accessing the microphone data.

- file **e_prox.c**

  Accessing proximity sensor of e-puck with timer 1.

- file **e_prox.h**

  Accessing proximity sensor of e-puck with timer 1.

- file **e_prox_timer2.c**

  Control proximity sensor of e-puck with timer2.

6.1.1 Detailed Description

6.1.2 Introduction

The microcontroller p30f6014A has a 12-bit Analog-to-Digital Converter (ADC). This package is built to manage this ADC.

The e-puck has three peripherals which take advantage from it.

- 1 3D accelerometer
- 8 proximity sensors
- 3 microphones

6.1.3 Package organization

This package is divided by two sub packages:

- The standard approach (files located in the "a_d" folder). This approach uses the timer1 (or timer2) to coordinate the ADC register acquisition.
- The more advanced approach (files located in the "a_d/advance_ad_scan" folder) uses the ADC interrupt to update the four **arrays** containing all the data of all the peripherals which use the ADC (**e_mic_scan** (p. 83) for the microphones, **e_acc_scan** (p. 76) for the accelerometer, **e_ambient_ir** (p. 92) and **e_ambient_and_reflected_ir** (p. 92) for the proximity sensors). In this approach, the acquisition is made automatically and always with the same delay. The functions specific to each module (**advance_ad_scan/e_acc.c** (p. 63), **advance_ad_scan/e_prox.c** (p. 90), **advance_ad_scan/e_micro.c** (p. 82)) are also more elaborates than the same one in the standard package.

Author:

Doc: Jonathan Besuchet

6.2 Bluetooth

Files

- file **e_bluetooth.c**

  Manage Bluetooth.

- file **e_bluetooth.h**

  Manage Bluetooth.

6.2.1 Detailed Description

6.2.2 Introduction

This package contains all the ressources you need to control the bluetooth device you have on your e-puck AS MASTER. If you only want to use the bluetooth AS SLAVE, the uart library is enough (the connection to a master look like a standard uart connection).

The bluetooth device is connected on the uart1.

Generally, the bluetooth modules have a bluetooth class device which identify them as PC, mobile, mouse, ... The e-puck bluetooth module has the default device class number, it's 000.

To learn more about the e-puck's bluetooth device see the documentation of the LMX9820A from National Semiconductor (look at `http://www.national.com/mpf/LM/LMX9820A.html`) and look at the **UART** (p. 36) module.

Author:

Doc: Jonathan Besuchet

6.3 Camera fast two timers

Files

- file **e_calc.c**

  Calculate the timing for the camera (two timers).

- file **e_po3030k.h**

  PO3030k library header (two timers).

- file **e_timers.c**

  Manage camera's interrupts (two timers).

6.3.1 Detailed Description

6.3.2 Introduction

This driver expose most of the Po3030k camera interfaces. Some functions are usefull, some other not. But since this is not up to the driver to decide if a function is needed, I've exported almost all.

The architecture is quite simple. The driver keep a array where every known camera register is keept in memory. The configuration function only alter this array. When you call, for example **e_po3030k_config_cam()** (p. 121), nothing is written on the camera, but only in the internal register representation.

To effectively write the register, you must call **e_po3030k_write_cam_registers()** (p. 140). This is typically done after every configuration call and before acquire the first picture.

6.3.3 Default settings

The camera is, by default, configured with the followin settings:

- Automatic white balance control
- Automatic exposure control
- Automatic flicker detection (50Hz and 60Hz)

There is no default setting for the image size and color.

6.3.4 Performances

The maximum framerate (without doing anything else than acquiring the picture) vary with the subsampling and the color mode. Here are some framerates:

- Size: 640x480, Subsampling: 16x16, RGB565: 4.3 fps
- Size: 16x480, Subsampling: 16x16, RGB565: 4.3 fps
- Size: 480x16, Subsampling: 16x16: RGB565: 4.3fps
- Size: 64x64, Subsampling: 4x4, RGB565: 4.3 fps

- Size: 32x32, Subsampling: 2x2, RGB565: 2.6 fps
- Size: 16x16, No Subsampling, RGB565: 1.3 fps
- Size: 640x480, Subsampling: 16x16, GREYSCALE: 8.6 fps
- Size: 16x480, Subsampling: 16x16, GREYSCALE: 8.6 fps
- Size: 480x16, Subsampling: 16x16, GREYSCALE: 8.6 fps
- Size: 64x64, Subsampling: 4x4, GREYSCALE: 8.6 fps
- Size: 32x32, Subsampling: 2x2, GREYSCALE: 4.3 fps
- Size: 16x16, No subsampling, GREYSCALE: 2.2 fps

6.3.5 Important note

This driver is extremly sensible to interrupt latency, thus it use interrupt priority to be sure that the latencies are kepts low. The Timer4 and Timer5 interrupt priority are set at level 6 and interrupt nesting is enabled. The Timer4 interrupt use the "push.s" and "pop.s" instructions. You should not have any code using thoses two instructions when you use the camera. This include the _ISRFAST C macro. If you use them, some random and really hard to debug behavior will happen. You have been warned !

6.3.6 Examples

6.3.6.1 Basic example

```
#include "e_po3030k.h"

char buffer[2*40*40];
int main(void) {

        e_po3030k_init_cam();
        e_po3030k_config_cam((ARRAY_WIDTH -160)/2,(ARRAY_HEIGHT-160)/2,
                        160,160,4,4,RGB_565_MODE);
        e_po3030k_write_cam_registers();

        e_po3030k_launch_capture(buffer);
        while(!e_po3030k_is_img_ready());

        // buffer contain a 40*40 RGB picture now
        ( insert usefull code here )

        return 0;
}
```

This example tell de driver to aquire 160x160 pixel picture from the camera 4x subsampling, thus resulting with a 40x40 pixel. The buffer as a size of $40*40*2$ because RGB565 is a two bytes per pixel data format.

6.3.6.2 More advanced example

```
#include "e_po3030k.h"

char buffer[160*2];
int main(void) {
        e_po3030k_init_cam();
        e_po3030k_config_cam((ARRAY_WIDTH - 320)/2,(ARRAY_HEIGHT - 32)/2,
```

```
                    320,8,2,4,GREY_SCALE_MODE);
        e_po3030k_set_mirror(1,1);
        e_po3030ke_set_ref_exposure(100);

        e_po3030k_write_cam_registers();

        e_po3030k_launch_capture(buffer);
        while(!e_po3030k_is_img_ready());

        // Here buffer contain a 160x2 greyscale picture

        return 0;
}
```

This example configure the camera to aquire a 320x8 pixel picture, but subsampled 2x in width and 4x in heigth, thus resulting in a $160*2$ linear greyscale picture. It "emulate" a linear camera. This example tell the camera to to enable the vertical and horizontal mirror, and to set the average exposure to 100.

6.4 Camera slow three timers

Files

- file **e_calc.c**

  Calculate the timing for the camera (three timers).

- file **e_po3030k.h**

  PO3030k library header (three timers).

- file **e_registers.c**

  Manage po3030k registers (three timers).

- file **e_timers.c**

  Manage camera's interrupts (three timers).

6.4.1 Detailed Description

Warning:

This driver version is completly interrupt driven to syncronize with the camera. This slow down the acquisition a lot. You should use the other driver's version which syncronize with the camera only at each row.

6.4.2 Introduction

This driver expose most of the Po3030k camera interfaces. Some functions are usefull, some other not. But since this is not up to the driver to decide if a function is needed, I've exported almost all.

The architecture is quite simple. The driver keep a array where every known camera register is keept in memory. The configuration function only alter this array. When you call, for exemple **e_po3030k_config_cam()** (p. 121), nothing is written on the camera, but only in the internal register representation.

To effectively write the register, you must call **e_po3030k_write_cam_registers()** (p. 140). This is typically done after every configuration call and before acquire the first picture.

6.4.3 Default settings

The camera is, by default, configured with the followin settings:

- Automatic white balance control
- Automatic exposure control
- Automatic flicker detection (50Hz and 60Hz)

There is no default setting for the image size and color.

6.4.4 Performances

The maximum framerate (without doing anything else than acquiring the picture) vary with the subsampling and the color mode. Here are some framerates. Please use the other driver to have better performances :

- Size: 640x480, Subsampling: 16x16, RGB565: 0.6 fps
- Size: 64x64, Subsampling: 4x4, RGB565: 0.6 fps
- Size: 32x32, Subsampling: 2x2, RGB565: 0.3 fps
- Size: 16x16, No Subsampling, RGB565: 0.2 fps
- Size: 640x480, Subsampling: 16x16, GREYSCALE: 1.1 fps
- Size: 64x64, Subsampling: 4x4, GREYSCALE: 1.1 fps
- Size: 32x32, Subsampling: 2x2, GREYSCALE: 0.6 fps
- Size: 16x16, No subsampling, GREYSCALE: 0.3 fps

6.4.5 Exemples

6.4.5.1 Basic example

```
#include "e_po3030k.h"

char buffer[2*40*40];
int main(void) {

        e_po3030k_init_cam();
        e_po3030k_config_cam((ARRAY_WIDTH -160)/2,(ARRAY_HEIGHT-160)/2,
                        160,160,4,4,RGB_565_MODE);
        e_po3030k_write_cam_registers();

        e_po3030k_launch_capture(buffer);
        while(!e_po3030k_is_img_ready());

        // buffer contain a 40*40 RGB picture now
        ( insert usefull code here )

        return 0;
}
```

This exemple tell de driver to aquire 160x160 pixel picture from the camera 4x subsampling, thus resulting with a 40x40 pixel. The buffer as a size of $40*40*2$ because RGB565 is a two bytes per pixel data format.

6.4.5.2 More advanced example

```
#include "e_po3030k.h"

char buffer[160*2];
int main(void) {
        e_po3030k_init_cam();
        e_po3030k_config_cam((ARRAY_WIDTH - 320)/2,(ARRAY_HEIGHT - 32)/2,
                        320,8,2,4,GREY_SCALE_MODE);
        e_po3030k_set_mirror(1,1);
        e_po3030k_set_ref_exposure(100);
```

```
        e_po3030k_write_cam_registers();

        e_po3030k_launch_capture(buffer);
        while(!e_po3030k_is_img_ready());

        // Here buffer contain a 160x2 greyscale picture

        return 0;
}
```

This exemple configure the camera to aquire a 320x8 pixel picture, but subsampled 2x in width and 4x in heigth, thus resulting in a 160∗2 linear greyscale picture. It "emulate" a linear camera. This exemple tell the camera to to enable the vertical and horizontal mirror, and to set the average exposure to 100.

6.5 Sound

Files

- file **e_sound.c**

  Package to play basics sounds on the e-puck's speaker.
  For more info look at this: ***Sound*** *(p. 21).*

- file **e_sound.h**

  Package to play basics sounds on the e-puck's speaker.

6.5.1 Detailed Description

6.5.2 Introduction

The e-puck has got a speaker on his top. This package is made to take advantage of it, by playing little sample.

6.5.3 Package organization

The internals functions of this package are written in assembler, because of speed. The sound you can play is in the file codec/e_const_sound.s

The externals functions are located in the file **codec/e_sound.c** (p. 197). There are three functions: **e_init_sound(void)** (p. 199), **e_play_sound(int sound_offset, int sound_length)** (p. 200) and void **e_close_sound(void)** (p. 199). When you want to play a sound YOU HAVE TO call **e_init_sound(void)** (p. 199) at first, but only the first time.

To play a sound call **e_play_sound(int sound_offset, int sound_length)** (p. 200). This function takes two parameters, the first set the begining of the sound, the second set the length of the sound. In fact it works like this:

- The "sound" which is in the file codec/e_const_sound.s is placed somewhere in e-puck's memory.
- When you call the function **e_play_sound(int sound_offset, int sound_length)** (p. 200), the words are sent to the DCI one by one from the offset you have specified with the first parameter. The number of words sent are specified with the second parameter.

If you don't want to use the sound anymore call e_close_sound.

Warning:

If you call void **e_close_sound(void)** (p. 199), you have to recall e**e_init_sound(void)** (p. 199) to play sound again.

6.5.4 Sounds plage

In the file codec/e_const_sound.s there are 19044 words. Five sounds are organized as following [begining, length]:

0, 2112 : "haa"

2116, 1760 : "spaah"

3878, 3412 : "ouah"

7294, 3732 : "yaouh"

11028, 8016 : "wouaaaaaaaah"

Then if you want to play the "yaouh" sound from the begining to the end just write this: e_play_sound(7294, 3732);

A little exemple which plays the five sounds one by one.

```
#include <codec/e_sound.h>
#include <motor_led/e_init_port.h>

int main(void)
{
	e_init_port();
	e_init_sound();
	while(1)
	{
		long i;
		e_play_sound(0, 2112); //sound 1
		for(i=0; i<4000000; i++) {asm("nop");}
		e_play_sound(2116, 1760); //sound 2
		for(i=0; i<4000000; i++) {asm("nop");}
		e_play_sound(3878, 3412); //sound 3
		for(i=0; i<4000000; i++) {asm("nop");}
		e_play_sound(7294, 3732); //sound 4
		for(i=0; i<4000000; i++) {asm("nop");}
		e_play_sound(11028, 8016); //sound 5
		for(i=0; i<4000000; i++) {asm("nop");}
	}
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

6.6 LIS sensor turret

Files

- file **e_devantech.h**

  Devantech sensor of e-puck.

6.6.1 Detailed Description

6.6.2 Introduction

This module is made for an extention of the e-puck. No documentation is available for now.

6.7 Radio communication

Files

- file **ComModule.h**

  Radio communication.

6.7.1 Detailed Description

6.7.2 Introduction

This module is made for an extention of the e-puck. No documentation is available for now.

6.8 FFT

Files

- file **e_fft.c**

  Package to mange the FFT.

- file **e_fft.h**

  Package to mange the FFT.

- file **e_fft_utilities.h**

  Some fft features.

- file **e_input_signal.c**

  Allocate memory and initialize the sigCmpx array.

- file **e_twiddle_factors.c**

  The FFT factor from Microchip.

6.8.1 Detailed Description

6.8.2 Introduction

The fast fourier transform (FFT) is really usefull, especially to work with the microphones data. This package contains all you need to perform the FFT.

The dsPic has some specials instructions (MAC instructions) which are used here.

6.8.3 How it works

To do the FFT on your data, first make this:

- choose the size of the array in which the FFT will be done. You have to choose one of the following values: 64, 128, 256 or 512.
- put your choice in the **FFT_BLOCK_LENGTH** (p. 215) (it's in the file **e_fft.h** (p. 215)).

Then you just have

- to copy your data in the sigCmpx array with the **e_fast_copy(int∗ in_array, int∗ out_array, int size)** (p. 216) function
- to call the **e_doFFT_asm(fractcomplex∗ sigCmpx)** (p. 215) function

A little code to illustrate this.

```
        e_ad_scan_on();
        // waiting all the 512 data (here we scan the microphones)
 while(!e_ad_is_array_filled());
        e_ad_scan_off();
```

```
// We put the mean to zero
e_subtract_mean(e_mic_scan[0], FFT_BLOCK_LENGTH, LOG2_BLOCK_LENGTH);
// We copy the array of micro zero to the FFT array
e_fast_copy(e_mic_scan[0], (int*)sigCmpx, FFT_BLOCK_LENGTH);

// The result is saved => we can launch a new acquisition
e_ad_scan_on();
// Now we are doing the FFT
e_doFFT_asm(sigCmpx);
```

Author:

Doc: Jonathan Besuchet

6.9 I2C

Files

- file **e_I2C_master_module.c**

  Manage I2C basics.

- file **e_I2C_master_module.h**

  Manage I2C basics.

- file **e_I2C_protocol.c**

  Manage I2C protocole.

- file **e_I2C_protocol.h**

  Manage I2C protocole.

6.9.1 Detailed Description

6.9.2 Introduction

This software allows using the I2C hardware module on a DsPic30f60xx in a master mode for a single master system.

This module manage the I2C basics functions (low level I2C functions). They are made to perform the basics tasks like:

- initializing the I2C on the microcontroller (**e_i2c_init(void)** (p. 225))
- sending the Start bit (**e_i2c_start(void)** (p. 225))
- sending the Restart bit (**e_i2c_restart(void)** (p. 225))
- sending the Stop bit (**e_i2c_stop(void)** (p. 226))
- sending the acknowledgement bit (**e_i2c_ack(void)** (p. 224))
- writing or receiving a byte (**e_i2c_write(char byte)** (p. 226), **e_i2c_read(char *buf)** (p. 225))
- ...

6.9.3 Overview of I2C protocol

The I2C-bus supports any IC fabrication process (NMOS, CMOS, bipolar). Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognized by a unique address (whether it's a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device.

6.9.3.1 Master-slave relation

The I2C-bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. As masters are usually microcontrollers, let's consider the case of a data transfer between two microcontrollers connected to the I2C-bus.

This highlights the master-slave and receiver-transmitter relationships to be found on the I2C-bus. It should be noted that these relationships are not permanent, but only depend on the direction of data transfer at that time. The transfer of data would proceed as follows:

1) Suppose microcontroller A wants to send information to module B:

- microcontroller A (master), addresses module B (slave)
- microcontroller A (master-transmitter), sends data to module B (slave-receiver)
- microcontroller A terminates the transfer

2) If microcontroller A wants to receive information from module B:

- microcontroller A (master) addresses module B (slave)
- microcontroller A (master-receiver) receives data from module B (slave-transmitter)
- microcontroller A terminates the transfer.

Even in this case, the master (microcontroller A) generates the timing and terminates the transfer.

6.9.3.2 Start and Stop conditions

Within the procedure of the I2C-bus, unique situations arise which are defined as START (S) and STOP (P) conditions.

A HIGH to LOW transition on the SDA line while SCL is HIGH is one such unique case. This situation indicates a START condition.

A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition.

The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical). Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware.

6.9.3.3 Transfering data

Every byte put on the SDA line must be 8-bits long (char type).

Acknowledge:

Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse.

The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse. Of course, set-up and hold times must also be taken into account.

Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received.

When a slave doesn't acknowledge the slave address (for example, it's unable to receive or transmit because it's performing some real-time function), the data line must be left HIGH by the slave. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer. If a slave-receiver does acknowledge the slave address but, some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave generating the not-acknowledge on the first byte to follow. The slave leaves the data line HIGH and the master generates a STOP or a repeated START condition.

If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition.

6.9.4 Use I2C on e-puck

On the e-puck, the microcontroller is always the master.

6.9.4.1 Basics functions

The functions of the files **e_I2C_master_module.c** (p. 219) and **e_I2C_master_module.h** (p. 223) are low level I2C functions.

They are made to perform the basics tasks like:

- initializing the I2C on the microcontroller (**e_i2c_init(void)** (p. 225))
- sending the Start bit (**e_i2c_start(void)** (p. 225))
- sending the Restart bit (**e_i2c_restart(void)** (p. 225))
- sending the Stop bit (**e_i2c_stop(void)** (p. 226))
- sending the acknowledgement bit (**e_i2c_ack(void)** (p. 224))
- writing or receiving a byte (**e_i2c_write(char byte)** (p. 226), **e_i2c_read(char *buf)** (p. 225))
- ...

6.9.4.2 More developed functions

The functions of the files **e_I2C_protocol.c** (p. 227) and **e_I2C_protocol.h** (p. 229) are made to directly send or receive data from or to a specified slave.

6.9.5 Reference

For more information about I2C:

- `http://en.wikipedia.org/wiki/I%C2%B2C`
- `http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf`
- `http://tmd.havit.cz/Papers/I2C.pdf`

Author:

Doc: Jonathan Besuchet

6.10 Matlab communication

Files

- file **matlab.c**

  To communicate with matlab.

- file **matlab.h**

  To communicate with matlab.

6.10.1 Detailed Description

6.10.2 Introduction

This package contains all the ressources you need to communicate with matlab through bluetooth.

To make the communication possible, you have to follow this steps:

- Open matlab and set the default direcories to "...\library\matlab\matlab files\"
- Connect your e-puck to your PC with bluetooth
- Call the matlab function "OpenEpuck('COMX')"; X is the number of the port on which the e-puck is connected.

6.10.2.1 Sending data from matlab

If you want to send data from matlab you only have to call the matlab function "EpuckSendData(data, dataType)"

- data is the array of value you want to send;
- dataType is a string argument which can be 'int8' to send char or 'int16' to send int;

On the e-puck side, to receive the data, you have to call the appropiate function depending of the data type you will receive. For exemple if matlab send int data, you have to call: **e_receive_int_from_matlab(int *data, int array_size)** (p. 239).

6.10.2.2 Sending data from e-puck

Now if you want to send data from e-puck to matlab you have to call the function **e_send_int_to_matlab(int* data, int array_size)** (p. 240) (send int data) on e-puck side and call "EpuckGetData" on matlab side. This function make the data convertion automatically, so call it to receive both of 'char' or 'int' data.

See also:

Matlab communication (p. 31)

Author:

Doc: Jonathan Besuchet

6.11 Ports, motors and LEDs

Files

- file **e_agenda.c**
  Manage the agendas (timer2).
- file **e_agenda.h**
  Manage the agendas (timer2).
- file **e_led.c**
  Manage the LEDs with blinking possibility (timer2).
- file **e_led.h**
  Manage the LEDs with blinking possibility (timer2).
- file **e_motors.c**
  Manage the motors (with timer2).
- file **e_motors.h**
  Manage the motors (with timer2).
- file **e_motors_kml.c**
  Manage the motors (with timer2).
- file **e_remote_control.c**
  Manage the IR receiver module (timer2).
- file **e_remote_control.h**
  Manage the LEDs with blinking possibility (timer2).
- file **e_remote_control.h**
  Manage the LEDs with blinking possibility (timer2).
- file **e_agenda_fast.c**
  Manage the fast agendas (timer1, 2, 3).
- file **e_agenda_fast.h**
  Manage the fast agendas (timer1, 2, 3).
- file **e_led.c**
  Manage the LEDs with blinking possibility (timer1, 2, 3).
- file **e_led.h**
  Manage the LEDs with blinking possibility (timer1, 2, 3).
- file **e_motors.c**
  Manage the motors (with timer1, 2, 3).
- file **e_motors.h**

Manage the motors (with timer1, 2, 3).

- file **e_epuck_ports.h**

  Define all the usefull names corresponding of e-puck's hardware.

- file **e_init_port.c**

  Initialize the ports on standard configuration.

- file **e_init_port.h**

  Initialize the ports on standard configuration.

- file **e_led.c**

  Manage the LEDs.
  A little exemple for the LEDs (all the LEDs are blinking).

- file **e_led.h**

  Manage the LEDs.
  A little exemple for the LEDs (all the LEDs are blinking).

- file **e_motors.c**

  Manage the motors (with timer 4 and 5).

- file **e_motors.h**

  Manage the motors (with timer 4 and 5).

- file **e_motors_timer3.c**

  Initialize the ports on standard configuration.

- file **e_motors_timer3.c**

  Initialize the ports on standard configuration.

6.11.1 Detailed Description

6.11.2 Introduction

This package contains all the resources you need to control the ports, the motors, the LED and the IR receiver of the e-puck.

6.11.2.1 Ports

The standard port's name of the p30f6014A microcontroller is not explicit in the e-puck context, so we need to redefine these names to make them more user friendly.

This work is made in the file: **e_epuck_ports.h** (p. 323).

6.11.2.2 Motors

The e-puck has two step by step motors called MOTOR1 (left) and MOTOR2 (right). To control the changing phase's sequence of these motors we need to use timers. Four possibilities are offered to you:

- standard: we use the timer4 for MOTOR2 and timer5 for MOTOR1. This solution is exploited by the file library\motor_led\e_motors.c.
- one timer standard: we use the timer3 for both MOTOR1 and MOTOR2. This solution is exploited by the file library\motor_led\e_motors_timer3.c
- advance one timer: we use the timer2 for both MOTOR1 and MOTOR2, but this time the mechanism work on the agenda method (see below or **e_agenda.h** (p. 245) for more information about agenda). This solution is exploited by the file library\motor_led\advance_one_timer\e_motors.c.
- fast agenda: we use the timer1,2,3 for both MOTOR1 and MOTOR2, but this time the mechanism work on the fast_agenda method (see below or **e_agenda_fast.h** (p. 318) for more information about fast_agenda). This solution is exploited by the file library\motor_led\advance_one_timer\fast_agenda\e_motors.c.

6.11.2.3 LED

The e-puck has 8 reds LEDs, a front LED and a body LED. All the functions needed to control these LEDs are in the file library\motor_led\e_led.c. This file is made for basics use. If you want blinking functions you have to work with these following files: library\motor_led\advance_one_timer\e_led.c or library\motor_led\advance_one_timer\fast_agenda\e_led.c. In the case you will work with agenda solution (see below or **e_agenda.h** (p. 245) or **e_agenda_fast.h** (p. 318) for more information about agenda or fast agenda).

6.11.2.4 IR remote

The e-puck has a IR receptor. To control this receptor look at this file: library\motor_led\advance_one_timer\e_remote_control.c.

Warning:

The IR remote uses the agenda solution, then it use timer2 (see below or **e_agenda.h** (p. 245) for more information about agenda).

6.11.3 Timer's problems

The p30f6014A microcontroller has five timers. The camera's package uses the timer4 and the timer5, so we can't exploit them to make the motors work when we want to use the camera. For this reason we can't use the standard solution above.

Warning:

If you are using the camera, you have to work with one of this three solutions explained above:

- one timer standard
- advance one timer
- fast agenda

6.11.4 Agenda solution

As we have seen, we can use the agenda solution to make the motors work.

So what is an agenda ?

An agenda is a structure which can launch a specific function (called callback function) with a given intervals. The agenda structure is made to work as chained list.

How it works ?

You create an agenda by specifying:

- the callback function you will call
- the delay between two calls
- the next element of the chained list

On each timer overflow all the agenda chained list is scanned and if an agenda in the list has reached the delay, then the callback function is called.

See also:

e_agenda.c (p. 241), **e_agenda_fast.c** (p. 313)

Author:

Doc: Jonathan Besuchet

6.12 UART

Files

- file **e_uart_char.h**
  Manage UART.

6.12.1 Detailed Description

6.12.2 Introduction

This package contains all the ressources you need to control the UART (universal asynchronous receiver transmitter). The microcontroller p30f6014A has two UART controller: UART1 and UART2.

Warning:

In this package, the functions are written in ASM. We have "e_init_uartX.s" file for the initializing functions, "e_uartX_rx.s" file for receiving data functions and "e_uartX_tx.s" file for transmitting data functions (X can be 1 or 2).
Even these files are written in ASM, you can call them by including the **e_uart_char.h** (p. 334) files in your C code (look at the exemple in **e_uart_char.h** (p. 334)).

6.12.3 Bluetooth

The e-puck has his bluetooth module connected on the uart1. Two ways are possibles when you work with bluetooth:

- you are the master, look at this: **Bluetooth** (p. 14)
- you are the slave.

When you are the slave, you can communicate with the master device exactely by the same way as you do to communicate through the uart. The bluetooth protocole is made to look like as transparent as possible. This is possible because the master initialize the communication and the bluetooth module answer automatically to create the connection. After that the connection was created, you can communicate with the master by using the uart protocole.

Author:

Doc: Jonathan Besuchet

Chapter 7

e-puck Directory Documentation

7.1 a_d/ Directory Reference

Directories

- directory **advance_ad_scan**

Files

- file **e_accelerometer.c**
  Accessing the accelerometer sensor data.
- file **e_accelerometer.h**
  Accessing the accelerometer sensor data.
- file **e_ad_conv.c**
  Module for the Analogic/Digital conversion.
- file **e_ad_conv.h**
  Module for the Analogic/Digital conversion.
- file **e_micro.c**
  Accessing the microphone data.
- file **e_micro.h**
  Accessing the microphone data.
- file **e_prox.c**
  Accessing proximity sensor of e-puck with timer 1.
- file **e_prox.h**
  Accessing proximity sensor of e-puck with timer 1.
- file **e_prox_timer2.c**
  Control proximity sensor of e-puck with timer2.

7.2 a_d/advance_ad_scan/ Directory Reference

Files

- file **e_acc.c**

  Accessing the accelerometer data.

- file **e_acc.h**

  Accessing the accelerometer data.

- file **e_ad_conv.c**

  Module for the advance Analogic/Digital conversion.

- file **e_ad_conv.h**

  Module for the advance Analogic/Digital conversion.

- file **e_micro.c**

  Accessing the microphone data.

- file **e_micro.h**

  Accessing the microphone data.

- file **e_prox.c**

  Accessing proximity sensor of e-puck.

- file **e_prox.h**

  Accessing proximity sensor of e-puck.

7.3 motor_led/advance_one_timer/ Directory Reference

Directories

- directory **fast_agenda**

Files

- file **e_agenda.c**

  Manage the agendas (timer2).

- file **e_agenda.h**

  Manage the agendas (timer2).

- file **e_led.c**

  Manage the LEDs with blinking possibility (timer2).

- file **e_led.h**

  Manage the LEDs with blinking possibility (timer2).

- file **e_motors.c**

  Manage the motors (with timer2).

- file **e_motors.h**

  Manage the motors (with timer2).

- file **e_motors_kml.c**

  Manage the motors (with timer2).

- file **e_remote_control.c**

  Manage the IR receiver module (timer2).

- file **e_remote_control.h**

  Manage the LEDs with blinking possibility (timer2).

7.4 bluetooth/ Directory Reference

Files

- file **e_bluetooth.c**
  Manage Bluetooth.
- file **e_bluetooth.h**
  Manage Bluetooth.

7.5 camera/ Directory Reference

Directories

- directory **fast_2_timer**
- directory **slow_3_timer**

7.6 codec/ Directory Reference

Files

- file **e_common.inc**
- file **e_sound.c**

  Package to play basics sounds on the e-puck's speaker.
  For more info look at this: ***Sound*** *(p. 21).*

- file **e_sound.h**

  Package to play basics sounds on the e-puck's speaker.

7.7 contrib/ Directory Reference

Directories

- directory **LIS_sensors_turret**
- directory **SWIS_com_module**

7.8 camera/fast_2_timer/ Directory Reference

Files

- file **e_calc.c**

  Calculate the timing for the camera (two timers).

- file **e_po3030k.h**

  PO3030k library header (two timers).

- file **e_registers.c**

  Manage po3030k registers (two timers).

- file **e_timers.c**

  Manage camera's interrupts (two timers).

7.9 motor_led/advance_one_timer/fast_agenda/ Directory Reference

Files

- file **e_agenda_fast.c**

  Manage the fast agendas (timer1, 2, 3).

- file **e_agenda_fast.h**

  Manage the fast agendas (timer1, 2, 3).

- file **e_led.c**

  Manage the LEDs with blinking possibility (timer1, 2, 3).

- file **e_led.h**

  Manage the LEDs with blinking possibility (timer1, 2, 3).

- file **e_motors.c**

  Manage the motors (with timer1, 2, 3).

- file **e_motors.h**

  Manage the motors (with timer1, 2, 3).

7.10 fft/ Directory Reference

Files

- file **e_fft.c**

  Package to mange the FFT.

- file **e_fft.h**

  Package to mange the FFT.

- file **e_fft_utilities.h**

  Some fft features.

- file **e_input_signal.c**

  Allocate memory and initialize the sigCmpx array.

- file **e_twiddle_factors.c**

  The FFT factor from Microchip.

7.11 I2C/ Directory Reference

Files

- file **e_I2C_master_module.c**
  Manage I2C basics.
- file **e_I2C_master_module.h**
  Manage I2C basics.
- file **e_I2C_protocol.c**
  Manage I2C protocole.
- file **e_I2C_protocol.h**
  Manage I2C protocole.

7.12 contrib/LIS_sensors_turret/ Directory Reference

Files

- file **e_devantech.c**
- file **e_devantech.h**

  Devantech sensor of e-puck.

- file **e_sensext.c**
- file **e_sensext.h**
- file **e_sharp.c**
- file **e_sharp.h**

7.13 matlab/ Directory Reference

Directories

- directory **matlab files**

Files

- file **matlab.c**

  To communicate with matlab.

- file **matlab.h**

  To communicate with matlab.

7.14 matlab/matlab files/ Directory Reference

Files

- file **CloseEpuck.m**
- file **EpuckFlush.m**
- file **EpuckGetData.m**
- file **EpuckSendData.m**
- file **OpenEpuck.m**

7.15 motor_led/ Directory Reference

Directories

- directory **advance_one_timer**

Files

- file **e_epuck_ports.h**

  Define all the usefull names corresponding of e-puck's hardware.

- file **e_init_port.c**

  Initialize the ports on standard configuration.

- file **e_init_port.h**

  Initialize the ports on standard configuration.

- file **e_led.c**

  Manage the LEDs.
  A little exemple for the LEDs (all the LEDs are blinking).

- file **e_led.h**

  Manage the LEDs.
  A little exemple for the LEDs (all the LEDs are blinking).

- file **e_motors.c**

  Manage the motors (with timer 4 and 5).

- file **e_motors.h**

  Manage the motors (with timer 4 and 5).

- file **e_motors_timer3.c**

  Initialize the ports on standard configuration.

7.16 camera/slow_3_timer/ Directory Reference

Files

- file **e_calc.c**

  Calculate the timing for the camera (three timers).

- file **e_po3030k.h**

  PO3030k library header (three timers).

- file **e_registers.c**

  Manage po3030k registers (three timers).

- file **e_timers.c**

  Manage camera's interrupts (three timers).

7.17 contrib/SWIS_com_module/ Directory Reference

Files

- file **ComModule.c**
- file **ComModule.h**

  Radio communication.

7.18 uart/ Directory Reference

Files

- file **e_epuck_ports.inc**
- file **e_uart_char.h**

  Manage UART.

Chapter 8

e-puck Data Structure Documentation

8.1 AgendaList Struct Reference

Manage the differents agendas lists.

`#include <e_agenda_fast.h>`

Data Fields

- char **motors**
- **Agenda** * **waiting**
- **Agenda** * **agendas** [3]
- unsigned char **timers_in_use** [3]
- unsigned **speed** [3]

8.1.1 Detailed Description

Manage the differents agendas lists.

We use the 3 first timers, then we need 3 pointers of the beginning of each Agenda chained list. We also have a pointer of the waiting Agenda chained list.

8.1.2 Field Documentation

8.1.2.1 char AgendaList::motors

8.1.2.2 Agenda* AgendaList::waiting

If an agenda's speed goes down to zero, we remove it from the list and add it to the waiting list

8.1.2.3 Agenda* AgendaList::agendas[3]

We use the 3 first timers

8.1.2.4 unsigned char AgendaList::timers_in_use[3]

Determine which one we use currently

8.1.2.5 unsigned AgendaList::speed[3]

Base speed 0.1 ms but use of multiplication

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

- motor_led/advance_one_timer/fast_agenda/**e_agenda_fast.h**

8.2 AgendaType Struct Reference

srtuct Agenda as chained list

`#include <e_agenda.h>`

Data Fields

- unsigned int **cycle**
- int **counter**
- char **activate**
- void(∗ **function**)(void)
- **Agenda** ∗ **next**
- **Agenda** ∗ **next**

8.2.1 Detailed Description

srtuct Agenda as chained list

The role of Agenda is to launch the pointed function when the member "counter" is greater than the member "cycle".

The member "activate" can be on=1 or off=0. When it is off, the counter don't increase.

This struct is designed to be used as chained list so we need a pointer to the next element.

8.2.2 Field Documentation

8.2.2.1 unsigned int AgendaType::cycle

length in 10e of ms of a cycle between two events

8.2.2.2 int AgendaType::counter

count the number of interrupts

8.2.2.3 char AgendaType::activate

can be on=1 or off=0

8.2.2.4 void(∗ AgendaType::function)(void) (void)

function called when counter > cycle,

Warning:

This function must have the following prototype: "void func(void)"

8.2.2.5 Agenda* AgendaType::next

pointer on the next agenda

8.2.2.6 Agenda* AgendaType::next

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

- motor_led/advance_one_timer/**e_agenda.h**
- motor_led/advance_one_timer/fast_agenda/**e_agenda_fast.h**

8.3 BtDevice Struct Reference

general struct for bluetooth device

`#include <e_bluetooth.h>`

Data Fields

- unsigned char **address** [6]
- unsigned char **class** [3]
- char **friendly_name** [20]

8.3.1 Detailed Description

general struct for bluetooth device

8.3.2 Field Documentation

8.3.2.1 unsigned char BtDevice::address[6]

Bluetooth MAC address

8.3.2.2 unsigned char BtDevice::class[3]

Bluetooth class of device

8.3.2.3 char BtDevice::friendly_name[20]

The friendly name of the bluetooth device

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

- bluetooth/**e_bluetooth.h**

8.4 BtEPuck Struct Reference

general struct for other e-puck

`#include <e_bluetooth.h>`

Data Fields

- unsigned char **address** [6]
- unsigned char **number** [5]

8.4.1 Detailed Description

general struct for other e-puck

8.4.2 Field Documentation

8.4.2.1 unsigned char BtEPuck::address[6]

e-puck's bluetooth MAC address

8.4.2.2 unsigned char BtEPuck::number[5]

e-puck's bluetooth PIN

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

- bluetooth/**e_bluetooth.h**

8.5 TypeAccRaw Struct Reference

struct to store the acceleration raw data in carthesian coord

`#include <e_acc.h>`

Data Fields

- int **acc_x**
- int **acc_y**
- int **acc_z**

8.5.1 Detailed Description

struct to store the acceleration raw data in carthesian coord

8.5.2 Field Documentation

8.5.2.1 int TypeAccRaw::acc_x

The acceleration on x axis

8.5.2.2 int TypeAccRaw::acc_y

The acceleration on y axis

8.5.2.3 int TypeAccRaw::acc_z

The acceleration on z axis

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

- a_d/advance_ad_scan/**e_acc.h**

8.6 TypeAccSpheric Struct Reference

struct to store the acceleration vector in spherical coord

`#include <e_acc.h>`

Data Fields

- float **acceleration**
- float **orientation**
- float **inclination**

8.6.1 Detailed Description

struct to store the acceleration vector in spherical coord

8.6.2 Field Documentation

8.6.2.1 float TypeAccSpheric::acceleration

lenght of the acceleration vector = intensity of the acceleration

8.6.2.2 float TypeAccSpheric::orientation

orientation of the acceleration vector in the horizontal plan, zero facing front

- 0 deg = inclination to the front (front part lower than rear part)
- 90 deg = inclination to the left (left part lower than right part)
- 180 deg = inclination to the rear (rear part lower than front part)
- 270 deg = inclination to the right (right part lower than left part)

8.6.2.3 float TypeAccSpheric::inclination

inclination angle with the horizontal plan

- 0 deg = e-puck horizontal
- 90 deg = e-puck vertical
- 180 deg = e-puck horizontal but up-side-down

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

- a_d/advance_ad_scan/**e_acc.h**

Chapter 9

e-puck File Documentation

9.1 a_d/advance_ad_scan/e_acc.c File Reference

Accessing the accelerometer data.

```
#include "math.h"
#include "e_acc.h"
#include "e_ad_conv.h"
#include "../../motor_led/e_epuck_ports.h"
#include "../../motor_led/advance_one_timer/e_led.h"
#include <stdlib.h>
```

Functions

- int **e_get_acc** (unsigned int captor)
  Read the last value of a given accelerator axes.
- int **e_get_acc_filtered** (unsigned int captor, unsigned int filter_size)
  Read the value of a channel, filtered by an averaging filter.
- void **calculate_acc_raw** (void)
  read the x, y, z values, apply an averaging filter and finaly subbstract the center values.
- void **calculate_acc_spherical** (void)
  calculate the intensity of the acceleration vector, and the Euler's angles
- void **e_acc_calibr** (void)
  initialize de ad converter and calculate the zero values
- **TypeAccSpheric e_read_acc_spheric** (void)
  calculate and return the accel. in spherical coord
- float **e_read_inclination** (void)
  calculate and return the inclination angle

- float **e_read_orientation** (void)

  calculate and return the orientation angle

- float **e_read_acc** (void)

  calculate and return the intensity of the acceleration

- **TypeAccRaw e_read_acc_xyz** (void)

  calculate and return acceleration on the x,y,z axis

- int **e_read_acc_x** (void)

  calculate and return acceleration on the x axis

- int **e_read_acc_y** (void)

  calculate and return acceleration on the y axis

- int **e_read_acc_z** (void)

  calculate and return acceleration on the z axis

- void **e_display_angle** (void)

  light the led according to the orientation angle

Variables

- int **e_acc_scan** [3][ACC_SAMP_NB]
- unsigned int **e_last_acc_scan_id**
- static int **angle_mem** = 0
- static int **centre_x** = 0
- static int **centre_y** = 0
- static int **centre_z** = 2000
- static int **acc_x**
- static int **acc_y**
- static int **acc_z**
- static float **acceleration**
- static float **orientation**
- static float **inclination**

9.1.1 Detailed Description

Accessing the accelerometer data.

The functions of this file are made to deal with the accelerometer data. You can know the magnitude, the orientation, the inclination, ... of the acceleration that the e-puck is enduring.

Two structures are used:

- **TypeAccSpheric** (p. 62) to store the acceleration data on sherical coordinates.
- **TypeAccRaw** (p. 61) to store the acceleration data on cartesian coordinates.

A little exemple to read the accelerator.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/advance_ad_scan/e_ad_conv.h>
#include <a_d/advance_ad_scan/e_acc.h>

int main(void)
{
       int z;
       e_init_port();
       e_init_ad_scan();
       while(1)
       {
               long i;
                       z = e_get_acc(2);
               if(z < 2100)    //LED4 on if e-puck is on the back
               {
                       LED0 = 0;
                       LED4 = 1;
               }
               else            //LED0 on if e-puck is on his wells
               {
                       LED0 = 1;
                       LED4 = 0;
               }
               for(i=0; i<100000; i++) { asm("nop"); }
               }
 }
```

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

9.1.2 Function Documentation

9.1.2.1 void calculate_acc_raw (void)

read the x, y, z values, apply an averaging filter and finaly subbstract the center values.

9.1.2.2 void calculate_acc_spherical (void)

calculate the intensity of the acceleration vector, and the Euler's angles

9.1.2.3 void e_acc_calibr (void)

initialize de ad converter and calculate the zero values

It reads two times the average_size to avoid edge effects then it reads 100 values and average them to initiate the "zero" value of the accelerometer

9.1.2.4 void e_display_angle (void)

light the led according to the orientation angle

9.1.2.5 int e_get_acc (unsigned int *captor*)

Read the last value of a given accelerator axes.

Parameters:

captor ID of the AD channel to read (must be 0 = x, 1 = y or 2 = z)

Returns:

value filtered channel's value

9.1.2.6 int e_get_acc_filtered (unsigned int *captor*, unsigned int *filter_size*)

Read the value of a channel, filtered by an averaging filter.

Parameters:

captor ID of the AD channel to read (must be 0 to 2)
filter_size size of the filter (must be between 1 and SAMPLE_NUMBER)

Returns:

value filtered channel's value

9.1.2.7 float e_read_acc (void)

calculate and return the intensity of the acceleration

Returns:

intensity of the acceleration vector

9.1.2.8 TypeAccSpheric e_read_acc_spheric (void)

calculate and return the accel. in spherical coord

Returns:

acceleration in spherical coord

See also:

TypeAccSpheric (p. 62)

9.1.2.9 int e_read_acc_x (void)

calculate and return acceleration on the x axis

Returns:

acceleration on the x axis

9.1.2.10 TypeAccRaw e_read_acc_xyz (void)

calculate and return acceleration on the x,y,z axis

Returns:

acceleration on the x,y,z axis

See also:

TypeAccRaw (p. 61)

9.1.2.11 int e_read_acc_y (void)

calculate and return acceleration on the y axis

Returns:

acceleration on the y axis

9.1.2.12 int e_read_acc_z (void)

calculate and return acceleration on the z axis

Returns:

acceleration on the z axis

9.1.2.13 float e_read_inclination (void)

calculate and return the inclination angle

Returns:

inclination angle of the robot

9.1.2.14 float e_read_orientation (void)

calculate and return the orientation angle

Returns:

orientation of the accel vector

9.1.3 Variable Documentation

9.1.3.1 int acc_x [static]

9.1.3.2 int acc_y [static]

9.1.3.3 int acc_z [static]

9.1.3.4 float acceleration [static]

9.1.3.5 int angle_mem = 0 [static]

9.1.3.6 int centre_x = 0 [static]

9.1.3.7 int centre_y = 0 [static]

9.1.3.8 int centre_z = 2000 [static]

9.1.3.9 int e_acc_scan[3][ACC_SAMP_NB]

Array to store the acc values

9.1.3.10 unsigned int e_last_acc_scan_id

9.1.3.11 float inclination [static]

9.1.3.12 float orientation [static]

9.2 a_d/advance_ad_scan/e_acc.h File Reference

Accessing the accelerometer data.

Data Structures

- struct **TypeAccSpheric**

  struct to store the acceleration vector in spherical coord

- struct **TypeAccRaw**

  struct to store the acceleration raw data in carthesian coord

Defines

- #define **CST_RADIAN** (180.0/3.1415)
- #define **ANGLE_ERROR** 666.0
- #define **FILTER_SIZE** 5
- #define **ACCX_BUFFER** 0
- #define **ACCY_BUFFER** 1
- #define **ACCZ_BUFFER** 2

Functions

- int **e_get_acc** (unsigned int captor)

  Read the last value of a given accelerator axes.

- int **e_get_acc_filtered** (unsigned int captor, unsigned int filter_size)

  Read the value of a channel, filtered by an averaging filter.

- **TypeAccSpheric e_read_acc_spheric** (void)

  calculate and return the accel. in spherical coord

- float **e_read_orientation** (void)

  calculate and return the orientation angle

- float **e_read_inclination** (void)

  calculate and return the inclination angle

- float **e_read_acc** (void)

  calculate and return the intensity of the acceleration

- **TypeAccRaw e_read_acc_xyz** (void)

  calculate and return acceleration on the x,y,z axis

- int **e_read_acc_x** (void)

  calculate and return acceleration on the x axis

- int **e_read_acc_y** (void)
  calculate and return acceleration on the y axis
- int **e_read_acc_z** (void)
  calculate and return acceleration on the z axis
- void **e_acc_calibr** (void)
  initialize de ad converter and calculate the zero values
- void **e_display_angle** (void)
  light the led according to the orientation angle

9.2.1 Detailed Description

Accessing the accelerometer data.

The functions of this file are made to deal with the accelerometer data. You can know the magnitude, the orientation, the inclination, ... of the acceleration that the e-puck is enduring.

Two structures are used:

- **TypeAccSpheric** (p. 62) to store the acceleration data on sherical coordinates.
- **TypeAccRaw** (p. 61) to store the acceleration data on cartesian coordinates.

A little exemple to read the accelerator.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/advance_ad_scan/e_ad_conv.h>
#include <a_d/advance_ad_scan/e_acc.h>

int main(void)
{
       int z;
       e_init_port();
       e_init_ad_scan();
       while(1)
       {
               long i;
                       z = e_get_acc(2);
               if(z < 2100)    //LED4 on if e-puck is on the back
               {
                       LED0 = 0;
                       LED4 = 1;
               }
               else            //LED0 on if e-puck is on his wells
               {
                       LED0 = 1;
                       LED4 = 0;
               }
               for(i=0; i<100000; i++) { asm("nop"); }
               }
 }
```

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

9.2.2 Define Documentation

9.2.2.1 #define ACCX_BUFFER 0

9.2.2.2 #define ACCY_BUFFER 1

9.2.2.3 #define ACCZ_BUFFER 2

9.2.2.4 #define ANGLE_ERROR 666.0

9.2.2.5 #define CST_RADIAN (180.0/3.1415)

9.2.2.6 #define FILTER_SIZE 5

9.2.3 Function Documentation

9.2.3.1 void e_acc_calibr (void)

initialize de ad converter and calculate the zero values

It reads two times the average_size to avoid edge effects then it reads 100 values and average them to initiate the "zero" value of the accelerometer

9.2.3.2 void e_display_angle (void)

light the led according to the orientation angle

9.2.3.3 int e_get_acc (unsigned int *captor*)

Read the last value of a given accelerator axes.

Parameters:

captor ID of the AD channel to read (must be 0 = x, 1 = y or 2 = z)

Returns:

value filtered channel's value

9.2.3.4 int e_get_acc_filtered (unsigned int *captor*, unsigned int *filter_size*)

Read the value of a channel, filtered by an averaging filter.

Parameters:

captor ID of the AD channel to read (must be 0 to 2)

filter_size size of the filter (must be between 1 and SAMPLE_NUMBER)

Returns:

value filtered channel's value

9.2.3.5 float e_read_acc (void)

calculate and return the intensity of the acceleration

Returns:

intensity of the acceleration vector

9.2.3.6 TypeAccSpheric e_read_acc_spheric (void)

calculate and return the accel. in spherical coord

Returns:

acceleration in spherical coord

See also:

TypeAccSpheric (p. 62)

9.2.3.7 int e_read_acc_x (void)

calculate and return acceleration on the x axis

Returns:

acceleration on the x axis

9.2.3.8 TypeAccRaw e_read_acc_xyz (void)

calculate and return acceleration on the x,y,z axis

Returns:

acceleration on the x,y,z axis

See also:

TypeAccRaw (p. 61)

9.2.3.9 int e_read_acc_y (void)

calculate and return acceleration on the y axis

Returns:

acceleration on the y axis

9.2.3.10 int e_read_acc_z (void)

calculate and return acceleration on the z axis

Returns:

acceleration on the z axis

9.2.3.11 float e_read_inclination (void)

calculate and return the inclination angle

Returns:

inclination angle of the robot

9.2.3.12 float e_read_orientation (void)

calculate and return the orientation angle

Returns:

orientation of the accel vector

9.3 a_d/advance_ad_scan/e_ad_conv.c File Reference

Module for the advance Analogic/Digital conversion.

```
#include "./../../motor_led/e_epuck_ports.h"
#include "e_ad_conv.h"
```

Functions

- void **e_init_ad_scan** (unsigned char only_micro)

  Initialize all the A/D register needed.

- void **__attribute__** ((__interrupt__, auto_psv))

  Save the AD buffer registers in differents arrays.

- unsigned char **e_ad_is_acquisition_completed** (void)

  To know if the ADC acquisitionn is completed.

- unsigned char **e_ad_is_array_filled** (void)

  To know if the ADC acquisitionn of microphone only is completed.

- void **e_ad_scan_on** (void)

  Enable the ADC conversion.

- void **e_ad_scan_off** (void)

  Disable the ADC conversion.

Variables

- int **e_mic_scan** [3][MIC_SAMP_NB]
- int **e_acc_scan** [3][ACC_SAMP_NB]
- unsigned int **e_last_mic_scan_id** = 0
- unsigned int **e_last_acc_scan_id** = 0
- int **e_ambient_ir** [8]
- int **e_ambient_and_reflected_ir** [8]
- static unsigned char **is_ad_acquisition_completed** = 0
- static unsigned char **is_ad_array_filled** = 0
- static unsigned char **micro_only** = 0

9.3.1 Detailed Description

Module for the advance Analogic/Digital conversion.

Author:

Code: Francesco Mondada, Michael-Bonani & Borter Jean-Joel
Doc: Jonathan Besuchet

9.3.2 Function Documentation

9.3.2.1 void __attribute__ ((__interrupt__, auto_psv))

Save the AD buffer registers in differents arrays.

Interrupt from timer3.

Interrupt from timer2.

Parse the chained list of agenda.

Increment counter only.

Check if agenda has to be treated according to the cycle value and current counter value.

Do it for number of cycle positive or null.

Check if a service has to be activated.

9.3.2.2 unsigned char e_ad_is_acquisition_completed (void)

To know if the ADC acquisitionn is completed.

Returns:

0 if the new acquisition is not made, 1 if completed.

9.3.2.3 unsigned char e_ad_is_array_filled (void)

To know if the ADC acquisitionn of microphone only is completed.

Returns:

0 if the new acquisition is not made, 1 if completed.

9.3.2.4 void e_ad_scan_off (void)

Disable the ADC conversion.

9.3.2.5 void e_ad_scan_on (void)

Enable the ADC conversion.

9.3.2.6 void e_init_ad_scan (unsigned char *only_micro*)

Initialize all the A/D register needed.

Set up the different ADC register to process the AD conversion by scanning the used AD channels. Each value of the channels will be stored in a different AD buffer register and an inturrupt will occure at the end of the scan.

Parameters:

only_micro Put MICRO_ONLY to use only the three microphones at 33kHz. Put ALL_ADC to use all the stuff using the ADC.

9.3.3 Variable Documentation

9.3.3.1 int e_acc_scan[3][ACC_SAMP_NB]

Array to store the acc values

9.3.3.2 int e_ambient_and_reflected_ir[8]

Array to store the light when IR led is on

9.3.3.3 int e_ambient_ir[8]

Array to store the ambient light measurement

9.3.3.4 unsigned int e_last_acc_scan_id = 0

9.3.3.5 unsigned int e_last_mic_scan_id = 0

9.3.3.6 int e_mic_scan[3][MIC_SAMP_NB]

Array to store the mic values

9.3.3.7 unsigned char is_ad_acquisition_completed = 0 [static]

9.3.3.8 unsigned char is_ad_array_filled = 0 [static]

9.3.3.9 unsigned char micro_only = 0 [static]

9.4 a_d/e_ad_conv.c File Reference

Module for the Analogic/Digital conversion.

```
#include "../motor_led/e_epuck_ports.h"
```

Functions

- void **e_init_ad** (void)

  Initialize all the A/D register needed.

- int **e_read_ad** (unsigned int channel)

  Function to sample an AD channel.

9.4.1 Detailed Description

Module for the Analogic/Digital conversion.

Author:

Code: Lucas Meier & Francesco Mondada, Michael Bonami
Doc: Jonathan Besuchet

9.4.2 Function Documentation

9.4.2.1 void e_init_ad (void)

Initialize all the A/D register needed.

9.4.2.2 int e_read_ad (unsigned int *channel*)

Function to sample an AD channel.

Parameters:

channel The A/D channel you want to sample Must be between 0 to 15

Returns:

The sampled value on the specified channel

9.5 a_d/advance_ad_scan/e_ad_conv.h File Reference

Module for the advance Analogic/Digital conversion.

Defines

- #define **MIC_SAMP_FREQ** 16384.0
- #define **ACC_PROX_SAMP_FREQ** 256.0
- #define **PULSE_LENGHT** 0.0003
- #define **ACC_PROX_PERIOD** (int)(MIC_SAMP_FREQ/ACC_PROX_SAMP_FREQ)
- #define **PULSE_PERIOD** (int)(PULSE_LENGHT∗MIC_SAMP_FREQ)
- #define **ADCS_3_CHAN** (int)(2.0∗FCY/(MIC_SAMP_FREQ∗(14+1)∗3)-1)
- #define **ADCS_5_CHAN** (int)(2.0∗FCY/(MIC_SAMP_FREQ∗(14+1)∗5)-1)
- #define **ADCS_6_CHAN** (int)(2.0∗FCY/(MIC_SAMP_FREQ∗(14+1)∗6)-1)
- #define **MIC_SAMP_NB** 100
- #define **ACC_SAMP_NB** 50
- #define **MICRO_ONLY** 1
- #define **ALL_ADC** 0

Functions

- void **e_init_ad_scan** (unsigned char only_micro)

  Initialize all the A/D register needed.

- unsigned char **e_ad_is_array_filled** (void)

  To know if the ADC acquisitionn of microphone only is completed.

- unsigned char **e_ad_is_acquisition_completed** (void)

  To know if the ADC acquisitionn is completed.

- void **e_ad_scan_on** (void)

  Enable the ADC conversion.

- void **e_ad_scan_off** (void)

  Disable the ADC conversion.

9.5.1 Detailed Description

Module for the advance Analogic/Digital conversion.

The advance converter module is set to operate by itself. It uses the ADC interrupt to launch the acquisitions. Then no timer is needed.

The data sampled are stored in the corresponding array:

- **e_mic_scan** (p. 83)[3][MIC_SAMP_NB] Array to store the mic values
- **e_acc_scan** (p. 76)[3][ACC_SAMP_NB] Array to store the acc values
- **e_ambient_ir** (p. 92)[8] Array to store ambient light measurement

- **e_ambient_and_reflected_ir** (p. 92)[8] Array to store ambient and reflected light measurement

In all the files of this module (**e_acc.c** (p. 63), e_micro.c, e_prox.c), these arrays are declared has "extern". In this way we can access the arrays for exemple like this (function e_get_acc from **e_acc.c** (p. 63)):

```
extern int e_acc_scan[3][ACC_SAMP_NB];

int e_get_acc(unsigned int captor)
{
       if (captor < 3)
              return (e_acc_scan[captor][e_last_acc_scan_id]);
       else
              return((int)ANGLE_ERROR);
}
```

Author:

Code: Francesco Mondada, Michael-Bonani & Borter Jean-Joel
Doc: Jonathan Besuchet

9.5.2 Define Documentation

9.5.2.1 #define ACC_PROX_PERIOD (int)(MIC_SAMP_FREQ/ACC_PROX_SAMP_FREQ)

9.5.2.2 #define ACC_PROX_SAMP_FREQ 256.0

9.5.2.3 #define ACC_SAMP_NB 50

9.5.2.4 #define ADCS_3_CHAN (int)(2.0∗FCY/(MIC_SAMP_FREQ∗(14+1)∗3)-1)

9.5.2.5 #define ADCS_5_CHAN (int)(2.0∗FCY/(MIC_SAMP_FREQ∗(14+1)∗5)-1)

9.5.2.6 #define ADCS_6_CHAN (int)(2.0∗FCY/(MIC_SAMP_FREQ∗(14+1)∗6)-1)

9.5.2.7 #define ALL_ADC 0

9.5.2.8 #define MIC_SAMP_FREQ 16384.0

9.5.2.9 #define MIC_SAMP_NB 100

9.5.2.10 #define MICRO_ONLY 1

9.5.2.11 #define PULSE_LENGHT 0.0003

9.5.2.12 #define PULSE_PERIOD (int)(PULSE_LENGHT∗MIC_SAMP_FREQ)

9.5.3 Function Documentation

9.5.3.1 unsigned char e_ad_is_acquisition_completed (void)

To know if the ADC acquisitionn is completed.

Returns:

0 if the new acquisition is not made, 1 if completed.

9.5.3.2 unsigned char e_ad_is_array_filled (void)

To know if the ADC acquisitionn of microphone only is completed.

Returns:

0 if the new acquisition is not made, 1 if completed.

9.5.3.3 void e_ad_scan_off (void)

Disable the ADC conversion.

9.5.3.4 void e_ad_scan_on (void)

Enable the ADC conversion.

9.5.3.5 void e_init_ad_scan (unsigned char *only_micro*)

Initialize all the A/D register needed.

Set up the different ADC register to process the AD conversion by scanning the used AD channels. Each value of the channels will be stored in a different AD buffer register and an inturrupt will occure at the end of the scan.

Parameters:

only_micro Put MICRO_ONLY to use only the three microphones at 33kHz. Put ALL_ADC to use all the stuff using the ADC.

9.6 a_d/e_ad_conv.h File Reference

Module for the Analogic/Digital conversion.

Functions

- void **e_init_ad** (void)

  Initialize all the A/D register needed.

- int **e_read_ad** (unsigned int channel)

  Function to sample an AD channel.

9.6.1 Detailed Description

Module for the Analogic/Digital conversion.

Author:

Code: Lucas Meier & Francesco Mondada, Michael Bonami
Doc: Jonathan Besuchet

9.6.2 Function Documentation

9.6.2.1 void e_init_ad (void)

Initialize all the A/D register needed.

9.6.2.2 int e_read_ad (unsigned int *channel*)

Function to sample an AD channel.

Parameters:

channel The A/D channel you want to sample Must be between 0 to 15

Returns:

The sampled value on the specified channel

9.7 a_d/advance_ad_scan/e_micro.c File Reference

Accessing the microphone data.

```
#include "p30f6014A.h"
#include "e_ad_conv.h"
#include "../../motor_led/e_epuck_ports.h"
```

Functions

- int **e_get_micro** (unsigned int micro_id)
  Get the last value of a given micro.
- int **e_get_micro_average** (unsigned int micro_id, unsigned int filter_size)
  Get the average on a given number of sample from a micro.
- int **e_get_micro_volume** (unsigned int micro_id)
  Get the difference between the highest and lowest sample.
- void **e_get_micro_last_values** (int micro_id, int *result, unsigned samples_nb)
  Write to a given array, the last values of one micro.

Variables

- int **e_mic_scan** [3][MIC_SAMP_NB]
- unsigned int **e_last_mic_scan_id**

9.7.1 Detailed Description

Accessing the microphone data.

The functions of this file are made to deal with the microphones data. You can simply get the current value of a given microphone. You can know the volume of noise that the e-puck is enduring. You can average the signal with a specified size.

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

9.7.2 Function Documentation

9.7.2.1 int e_get_micro (unsigned int *micro_id*)

Get the last value of a given micro.

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

Returns:

result last value of the micro

9.7.2.2 int e_get_micro_average (unsigned int *micro_id*, unsigned int *filter_size*)

Get the average on a given number of sample from a micro.

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

filter_size number of sample to average

Returns:

result last value of the micro

9.7.2.3 void e_get_micro_last_values (int *micro_id*, int * *result*, unsigned *samples_nb*)

Write to a given array, the last values of one micro.

Write to a given array, the last values of one micro. The values are stored with the last one first, and the older one at the end of the array.

[t][t-1][t-2][t-3]...[t-(samples_nb-1)][t-samples_nb]

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

∗***result*** pointer on the result array

samples_nb size of the result array (must be between 1 and **MIC_SAMP_NB** (p. 79))

9.7.2.4 int e_get_micro_volume (unsigned int *micro_id*)

Get the difference between the highest and lowest sample.

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

Returns:

result volume

9.7.3 Variable Documentation

9.7.3.1 unsigned int e_last_mic_scan_id

9.7.3.2 int e_mic_scan[3][MIC_SAMP_NB]

Array to store the mic values

9.8 a_d/e_micro.c File Reference

Accessing the microphone data.

```
#include "e_ad_conv.h"
#include "./../motor_led/e_epuck_ports.h"
#include "e_micro.h"
```

Functions

- void **e_init_micro** (void)

  Init the microphone A/D converter.

- void **e_get_micro** (int ∗m0, int ∗m1, int ∗m2)

  To get the m0, m1, m2 microphones's values.

9.8.1 Detailed Description

Accessing the microphone data.

A little exemple which takes the volume of micro1 and if the sound level is more than 2000. The LED1 is turned on.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/e_micro.h>

int main(void)
{
	int m1, m2, m3;
	e_init_port();
	e_init_micro();
	while(1)
	{
		long i;
		e_get_micro(&m1, &m2, &m3);
		if(m1 < 2000)
			LED0 = 0;
		else
			LED0 = 1;
		for(i=0; i<100000; i++) { asm("nop"); }
	}
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.8.2 Function Documentation

9.8.2.1 void e_get_micro (int ∗ *m0*, int ∗ *m1*, int ∗ *m2*)

To get the m0, m1, m2 microphones's values.

Parameters:

m0 A pointer to store the m0 analogic value

m1 A pointer to store the m1 analogic value

m2 A pointer to store the m2 analogic value

9.8.2.2 void e_init_micro (void)

Init the microphone A/D converter.

Warning:

Must be called before starting using microphone

9.9 a_d/advance_ad_scan/e_micro.h File Reference

Accessing the microphone data.

Defines

- #define **MIC0_BUFFER** 0
- #define **MIC1_BUFFER** 1
- #define **MIC2_BUFFER** 2

Functions

- int **e_get_micro** (unsigned int micro_id)

  Get the last value of a given micro.

- int **e_get_micro_average** (unsigned int micro_id, unsigned int filter_size)

  Get the average on a given number of sample from a micro.

- int **e_get_micro_volume** (unsigned int micro_id)

  Get the difference between the highest and lowest sample.

9.9.1 Detailed Description

Accessing the microphone data.

The functions of this file are made to deal with the microphones data. You can simply get the current value of a given microphone. You can know the volume of noise that the e-puck is enduring. You can average the signal with a specified size.

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

9.9.2 Define Documentation

9.9.2.1 #define MIC0_BUFFER 0

9.9.2.2 #define MIC1_BUFFER 1

9.9.2.3 #define MIC2_BUFFER 2

9.9.3 Function Documentation

9.9.3.1 int e_get_micro (unsigned int *micro_id*)

Get the last value of a given micro.

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

Returns:

result last value of the micro

9.9.3.2 int e_get_micro_average (unsigned int *micro_id*, unsigned int *filter_size*)

Get the average on a given number of sample from a micro.

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

filter_size number of sample to average

Returns:

result last value of the micro

9.9.3.3 int e_get_micro_volume (unsigned int *micro_id*)

Get the difference between the highest and lowest sample.

Parameters:

micro_id micro's ID (0, 1, or 2) (use **MIC0_BUFFER** (p. 86), **MIC1_BUFFER** (p. 86) , **MIC2_-BUFFER** (p. 86) defined in e_micro.h)

Returns:

result volume

9.10 a_d/e_micro.h File Reference

Accessing the microphone data.

Functions

- void **e_init_micro** (void)

  Init the microphone A/D converter.

- void **e_get_micro** (int ∗m1, int ∗m2, int ∗m3)

  To get the m0, m1, m2 microphones's values.

9.10.1 Detailed Description

Accessing the microphone data.

A little exemple which takes the volume of micro1 and if the sound level is more than 2000. The LED1 is turned on.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/e_micro.h>

int main(void)
{
       int m1, m2, m3;
       e_init_port();
       e_init_micro();
       while(1)
       {
               long i;
               e_get_micro(&m1, &m2, &m3);
               if(m1 < 2000)
                       LED0 = 0;
               else
                       LED0 = 1;
               for(i=0; i<100000; i++) { asm("nop"); }
       }
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.10.2 Function Documentation

9.10.2.1 void e_get_micro (int ∗ *m0*, int ∗ *m1*, int ∗ *m2*)

To get the m0, m1, m2 microphones's values.

Parameters:

m0 A pointer to store the m0 analogic value

m1 A pointer to store the m1 analogic value

m2 A pointer to store the m2 analogic value

9.10.2.2 void e_init_micro (void)

Init the microphone A/D converter.

Warning:

Must be called before starting using microphone

9.11 a_d/advance_ad_scan/e_prox.c File Reference

Accessing proximity sensor of e-puck.

```
#include "e_ad_conv.h"
#include "../../motor_led/e_epuck_ports.h"
#include "e_prox.h"
```

Functions

- void **e_calibrate_ir** ()

  To calibrate your ir sensor.

- int **e_get_prox** (unsigned int sensor_number)

  To get the analogic proxy sensor value of a specific ir sensor.

- int **e_get_calibrated_prox** (unsigned int sensor_number)

  To get the calibrated value of the ir sensor.

- int **e_get_ambient_light** (unsigned int sensor_number)

  To get the analogic ambient light value of a specific ir sensor.

Variables

- int **e_ambient_ir** [8]
- int **e_ambient_and_reflected_ir** [8]
- static int **init_value_ir** [8] = {0,0,0,0,0,0,0,0}

9.11.1 Detailed Description

Accessing proximity sensor of e-puck.

The functions of this file are made to deal with the proximitiy data. You can know the value of the ambient light detected by the sensor. You can estimate the distance between the e-puck and an obstacle by using **e_get_prox(unsigned int sensor_number)** (p. 107) function.

A little exemple which turn the LED0 when an obstacle is detected by the proximity sensor number 0.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/advance_ad_scan/e_prox.h>
#include <a_d/advance_ad_scan/e_ad_conv.h>

int main(void)
{
       int proxy0;
       e_init_port();
       e_init_ad_scan();
       while(1)
       {
              long i;
```

```
                proxy0 = e_get_prox(0);
                if(proxy0 < 1000)
                        LED0 = 0;
                else
                        LED0 = 1;
                for(i=0; i<100000; i++) { asm("nop"); }
        }
}
```

Author:

Code: Lucas Meier
Doc: Jonathan Besuchet

9.11.2 Function Documentation

9.11.2.1 void e_calibrate_ir ()

To calibrate your ir sensor.

Warning:

Call this function one time before calling e_get_calibrated_prox

9.11.2.2 int e_get_ambient_light (unsigned int *sensor_number*)

To get the analogic ambient light value of a specific ir sensor.

This function return the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.11.2.3 int e_get_calibrated_prox (unsigned int *sensor_number*)

To get the calibrated value of the ir sensor.

This function return the calbrated analogic value of the ir sensor.

Warning:

Befroe using this function you have to calibrate your ir sensor (only one time) by calling **e_calibrate_ir()** (p. 91).

Parameters:

sensor_number The proxy sensor's number that you want the calibrated value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.11.2.4 int e_get_prox (unsigned int *sensor_number*)

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.11.3 Variable Documentation

9.11.3.1 int e_ambient_and_reflected_ir[8]

Array to store the light when IR led is on

9.11.3.2 int e_ambient_ir[8]

Array to store the ambient light measurement

9.11.3.3 int init_value_ir[8] = {0,0,0,0,0,0,0,0} [static]

9.12 a_d/e_prox.c File Reference

Accessing proximity sensor of e-puck with timer 1.

```
#include "e_ad_conv.h"
#include "./../motor_led/e_epuck_ports.h"
#include "e_prox.h"
```

Functions

- void **init_tmr1** (void)
- void **__attribute__** ((interrupt, auto_psv, shadow))
- void **e_init_prox** (void)

  Init the proxymity sensor A/D converter and the timer1.

- void **e_stop_prox** (void)

  Stop the acquisition (stop timer1).

- int **e_get_prox** (unsigned int sensor_number)

  To get the analogic proxy sensor value of a specific sensor.

- int **e_get_ambient_light** (unsigned int sensor_number)

  To get the analogic ambient light value of a specific sensor.

Variables

- static int **ambient_ir** [8]
- static int **ambient_and_reflected_ir** [8]
- static int **reflected_ir** [8]

9.12.1 Detailed Description

Accessing proximity sensor of e-puck with timer 1.

The functions of this file are made to deal with the proximity sensor data. You can know the value of the ambient light detected by the sensor. You can estimate the distance between the e-puck and an obstacle by using **e_get_prox(unsigned int sensor_number)** (p. 107) function.

A little exemple which turn the LED0 when an obstacle is detected by the proximity sensor number 0.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/e_prox.h>

int main(void)
{
       int value;
       e_init_port();
       e_init_prox();
       while(1)
       {
```

```
                long i;
                value = e_get_prox(0);
                if(value > 1000)        //LED0 on if an obstacle is detected by proxy0
                        LED0 = 1;
                else
                        LED0 = 0;
                for(i=0; i<100000; i++) { asm("nop"); }
        }
}
```

Warning:

This module uses the timer1

Author:

Code: Lucas Meier & Francesco Mondada, Michael Bonani
Doc: Jonathan Besuchet

9.12.2 Function Documentation

9.12.2.1 void __attribute__ ((interrupt, auto_psv, shadow))

9.12.2.2 int e_get_ambient_light (unsigned int *sensor_number*)

To get the analogic ambient light value of a specific sensor.

To get the analogic ambient light value of a specific ir sensor.

This function retur the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.12.2.3 int e_get_prox (unsigned int *sensor_number*)

To get the analogic proxy sensor value of a specific sensor.

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.12.2.4 void e_init_prox (void)

Init the proxymity sensor A/D converter and the timer1.

Warning:

Must be called before starting using proximity sensor

9.12.2.5 void e_stop_prox (void)

Stop the acquisition (stop timer1).

9.12.2.6 void init_tmr1 (void)

9.12.3 Variable Documentation

9.12.3.1 int ambient_and_reflected_ir[8] [static]

9.12.3.2 int ambient_ir[8] [static]

9.12.3.3 int reflected_ir[8] [static]

9.13 a_d/advance_ad_scan/e_prox.h File Reference

Accessing proximity sensor of e-puck.

Functions

- void **e_calibrate_ir** ()

  To calibrate your ir sensor.

- int **e_get_prox** (unsigned int sensor_number)

  To get the analogic proxy sensor value of a specific ir sensor.

- int **e_get_calibrated_prox** (unsigned int sensor_number)

  To get the calibrated value of the ir sensor.

- int **e_get_ambient_light** (unsigned int sensor_number)

  To get the analogic ambient light value of a specific ir sensor.

9.13.1 Detailed Description

Accessing proximity sensor of e-puck.

The functions of this file are made to deal with the proximitiy data. You can know the value of the ambient light detected by the sensor. You can estimate the distance between the e-puck and an obstacle by using e_get_prox function.

A little exemple which turn the LED0 when an obstacle is detected by the proximity sensor number 0.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/advance_ad_scan/e_prox.h>
#include <a_d/advance_ad_scan/e_ad_conv.h>

int main(void)
{
	int proxy0;
	e_init_port();
	e_init_ad_scan();
	while(1)
	{
		long i;
		proxy0 = e_get_prox(0);
		if(proxy0 < 1000)
			LED0 = 0;
		else
			LED0 = 1;
		for(i=0; i<100000; i++) { asm("nop"); }
	}
}
```

Author:

Code: Lucas Meier
Doc: Jonathan Besuchet

9.13.2 Function Documentation

9.13.2.1 void e_calibrate_ir ()

To calibrate your ir sensor.

Warning:

Call this function one time before calling e_get_calibrated_prox

9.13.2.2 int e_get_ambient_light (unsigned int *sensor_number*)

To get the analogic ambient light value of a specific ir sensor.

This function return the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

To get the analogic ambient light value of a specific ir sensor.

This function retur the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.13.2.3 int e_get_calibrated_prox (unsigned int *sensor_number*)

To get the calibrated value of the ir sensor.

This function return the calbrated analogic value of the ir sensor.

Warning:

Befroe using this function you have to calibrate your ir sensor (only one time) by calling **e_calibrate_ir()** (p. 91).

Parameters:

sensor_number The proxy sensor's number that you want the calibrated value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.13.2.4 int e_get_prox (unsigned int *sensor_number*)

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.14 a_d/e_prox.h File Reference

Accessing proximity sensor of e-puck with timer 1.

Functions

- void **e_init_prox** (void)
  Init the proxymity sensor A/D converter and the timer1.
- void **e_stop_prox** (void)
  Stop the acquisition (stop timer1).
- int **e_get_prox** (unsigned int sensor_number)
  To get the analogic proxy sensor value of a specific ir sensor.
- int **e_get_ambient_light** (unsigned int sensor_number)
  To get the analogic ambient light value of a specific ir sensor.

9.14.1 Detailed Description

Accessing proximity sensor of e-puck with timer 1.

The functions of this file are made to deal with the proximity sensor data. You can know the value of the ambient light detected by the sensor. You can estimate the distance between the e-puck and an obstacle by using **e_get_prox(unsigned int sensor_number)** (p. 107) function.

A little exemple which turn the LED0 when an obstacle is detected by the proximity sensor number 0.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/e_prox.h>

int main(void)
{
	int value;
	e_init_port();
	e_init_prox();
	while(1)
	{
		long i;
		value = e_get_prox(0);
		if(value > 1000)	//LED0 on if an obstacle is detected by proxy0
			LED0 = 1;
		else
			LED0 = 0;
		for(i=0; i<100000; i++) { asm("nop"); }
	}
}
```

Warning:

This module uses the timer1

Author:

Code: Lucas Meier & Francesco Mondada, Michael Bonani
Doc: Jonathan Besuchet

9.14.2 Function Documentation

9.14.2.1 int e_get_ambient_light (unsigned int *sensor_number*)

To get the analogic ambient light value of a specific ir sensor.

This function return the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

To get the analogic ambient light value of a specific ir sensor.

This function retur the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.14.2.2 int e_get_prox (unsigned int *sensor_number*)

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.14.2.3 void e_init_prox (void)

Init the proxymity sensor A/D converter and the timer1.

Warning:

Must be called before starting using proximity sensor

Init the proxymity sensor A/D converter and the timer1.

Warning:

Must be called before starting using proximity sensor

9.14.2.4 void e_stop_prox (void)

Stop the acquisition (stop timer1).

9.15 a_d/e_accelerometer.c File Reference

Accessing the accelerometer sensor data.

```
#include "e_ad_conv.h"
#include "../motor_led/e_epuck_ports.h"
#include "e_accelerometer.h"
```

Functions

- void **e_init_acc** (void)

  Init the accelerometer A/D converter.

- void **e_get_acc** (int *x, int *y, int *z)

  To get the analogic x, y, z axis accelerations.

9.15.1 Detailed Description

Accessing the accelerometer sensor data.

A little exemple to read the accelerator.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/e_accelerometer.h>

int main(void)
{
	int x, y, z;
	e_init_port();
	e_init_acc();
	while(1)
	{
		long i;
		e_get_acc(&x, &y, &z);
		if(z < 2100)    //LED4 on if e-puck is on the back
		{
			LED0 = 0;
			LED4 = 1;
		}
		else            //LED0 on if e-puck is on his wells
		{
			LED0 = 1;
			LED4 = 0;
		}
		for(i=0; i<100000; i++) { asm("nop"); }
	}
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.15.2 Function Documentation

9.15.2.1 void e_get_acc (int * x, int * y, int * z)

To get the analogic x, y, z axis accelerations.

Parameters:

- ***x*** A pointer to store the analogic x acceleration
- ***y*** A pointer to store the analogic y acceleration
- ***z*** A pointer to store the analogic z acceleration

9.15.2.2 void e_init_acc (void)

Init the accelerometer A/D converter.

Warning:

Must be called before starting using accelerometer

9.16 a_d/e_accelerometer.h File Reference

Accessing the accelerometer sensor data.

Functions

- void **e_init_acc** (void)

  Init the accelerometer A/D converter.

- void **e_get_acc** (int *x, int *y, int *z)

  To get the analogic x, y, z axis accelerations.

9.16.1 Detailed Description

Accessing the accelerometer sensor data.

A little exemple to read the accelerator.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <a_d/e_accelerometer.h>

int main(void)
{
	int x, y, z;
	e_init_port();
	e_init_acc();
	while(1)
	{
		long i;
		e_get_acc(&x, &y, &z);
		if(z < 2100)    //LED4 on if e-puck is on the back
		{
			LED0 = 0;
			LED4 = 1;
		}
		else            //LED0 on if e-puck is on his wells
		{
			LED0 = 1;
			LED4 = 0;
		}
		for(i=0; i<100000; i++) { asm("nop"); }
	}
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.16.2 Function Documentation

9.16.2.1 void e_get_acc (int * x, int * y, int * z)

To get the analogic x, y, z axis accelerations.

Parameters:

x A pointer to store the analogic x acceleration

y A pointer to store the analogic y acceleration

z A pointer to store the analogic z acceleration

9.16.2.2 void e_init_acc (void)

Init the accelerometer A/D converter.

Warning:

Must be called before starting using accelerometer

9.17 a_d/e_prox_timer2.c File Reference

Control proximity sensor of e-puck with timer2.

```
#include "e_ad_conv.h"
#include "./../motor_led/e_epuck_ports.h"
#include "e_prox.h"
```

Functions

- void **init_tmr2** (void)
- void **__attribute__** ((interrupt, auto_psv, shadow))
- void **e_init_prox** (void)

  Init the proxymity sensor A/D converter and the timer2.

- void **e_stop_prox** (void)

  Stop the acquisition (stop timer2).

- int **e_get_prox** (unsigned int sensor_number)

  To get the analogic proxy sensor value of a specific sensor.

- int **e_get_ambient_light** (unsigned int sensor_number)

  To get the analogic ambient light value of a specific sensor.

Variables

- static int **ambient_ir** [8]
- static int **ambient_and_reflected_ir** [8]
- static int **reflected_ir** [8]

9.17.1 Detailed Description

Control proximity sensor of e-puck with timer2.

The functions of this file are made to deal with the proximitiy data. You can know the value of the ambient light detected by the sensor. You can estimate the distance between the e-puck and an obstacle by using **e_get_prox(unsigned int sensor_number)** (p. 107) function.

Warning:

The module uses the timer2

See also:

e_prox.c, e_prox.h to use the timer1

Author:

Code: Lucas Meier & Francesco Mondada, Michael Bonani, Xavier Raemy
Doc: Jonathan Besuchet

9.17.2 Function Documentation

9.17.2.1 void __attribute__ ((interrupt, auto_psv, shadow))

9.17.2.2 int e_get_ambient_light (unsigned int *sensor_number*)

To get the analogic ambient light value of a specific sensor.

To get the analogic ambient light value of a specific ir sensor.

This function retur the analogic value of the ambient light measurement.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.17.2.3 int e_get_prox (unsigned int *sensor_number*)

To get the analogic proxy sensor value of a specific sensor.

To get the analogic proxy sensor value of a specific ir sensor.

To estimate the proxymity of an obstacle, we do the following things:

- measure the ambient light
- turn on the IR led of the sensor
- measure the reflected light + ambient light
- calculate: reflected light = (reflected light + ambient light) - ambient light
- turn off the IR led of the sensor

The result value of this function is: reflected light. More this value is great, more the obsacle is near.

Parameters:

sensor_number The proxy sensor's number that you want the value. Must be between 0 to 7.

Returns:

The analogic value of the specified proxy sensor

9.17.2.4 void e_init_prox (void)

Init the proxymity sensor A/D converter and the timer2.

Init the proxymity sensor A/D converter and the timer1.

Warning:

Must be called before starting using proximity sensor

9.17.2.5 void e_stop_prox (void)

Stop the acquisition (stop timer2).

Stop the acquisition (stop timer1).

9.17.2.6 void init_tmr2 (void)

9.17.3 Variable Documentation

9.17.3.1 int ambient_and_reflected_ir[8] [static]

9.17.3.2 int ambient_ir[8] [static]

9.17.3.3 int reflected_ir[8] [static]

9.18 bluetooth/e_bluetooth.c File Reference

Manage Bluetooth.

```
#include "./../uart/e_uart_char.h"
#include "./../motor_led/e_epuck_ports.h"
#include "e_bluetooth.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"
```

Functions

- int **e_bt_find_epuck** (void)
  Try to find other e-puck.
- char **e_bt_connect_epuck** (void)
- int **e_bt_reset** (void)
  Reset the bluetooth module.
- char **e_bt_factory_reset** (void)
  Factory reset of the bluetooth module.
- char **e_bt_tranparent_mode** (void)
  Change to transparent mode.
- void **e_bt_exit_tranparent_mode** (void)
  Exit from the transparent mode.
- char **e_bt_read_local_pin_number** (char ∗PIN)
  Read the PIN number of this e-puck's bluetooth module.
- char **e_bt_read_local_name** (char ∗name)
  Read the name of this e-puck's bluetooth module.
- char **e_bt_write_local_pin_number** (char ∗PIN)
  Write the PIN number on this e-puck's bluetooth module.
- char **e_bt_write_local_name** (char ∗name)
  Write the name on this e-puck's bluetooth module.
- int **e_bt_inquiry** (struct **BtDevice** ∗device)
  Research all the accessible bluetooth devices.
- char **e_bt_get_friendly_name** (struct **BtDevice** ∗device)
  To get the friendly name of a bluetooth device.
- char **e_bt_etablish_SPP_link** (char ∗**address**)

Try to connect to another bluetooth device.

- char **e_bt_release_SPP_link** (void)
  Unconnect from the current bluetooth device.
- char **e_bt_send_SPP_data** (char *∗***data**, char datalength)
  Send data to the current bluetooth device.
- char **e_bt_list_local_paired_device** (void)
  Make a list of the bluetooth address of paired device.
- char **e_bt_remove_local_paired_device** (int j)
  Remove a paired bluetooth device.

Variables

- unsigned char **e_bt_local_paired_device** [6 ∗8]
- struct **BtDevice e_bt_present_device** [10]
- struct **BtEPuck e_bt_present_epuck** [10]
- char **local_bt_PIN** [4]

9.18.1 Detailed Description

Manage Bluetooth.

This module manage the Bluetooth device.

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.18.2 Function Documentation

9.18.2.1 char e_bt_connect_epuck (void)

9.18.2.2 char e_bt_etablish_SPP_link (char ∗ *address*)

Try to connect to another bluetooth device.

Parameters:

address A pointer on the device address which you want to connect

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.3 void e_bt_exit_tranparent_mode (void)

Exit from the transparent mode.

9.18.2.4 char e_bt_factory_reset (void)

Factory reset of the bluetooth module.

Warning:

use this function only if you are sure, your e-puck must be restarted, and renamed!!!

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.5 int e_bt_find_epuck (void)

Try to find other e-puck.

This function make global inguiry and check which device are e-puck, and list them in globales tables.

Returns:

number of e-puck found

See also:

e_bt_present_device (p. 120), **e_bt_present_epuck** (p. 120)

9.18.2.6 char e_bt_get_friendly_name (struct BtDevice * *device*)

To get the friendly name of a bluetooth device.

Parameters:

device A pointer on the device that you want the name

Returns:

bluetooth error if one occur, 0 otherwise

See also:

BtDevice (p. 59)

9.18.2.7 int e_bt_inquiry (struct BtDevice * *device*)

Research all the accessible bluetooth devices.

Parameters:

device A pointer to **BtDevice** (p. 59) to store all the caracteristics of each devices found

Returns:

the number of device found

See also:

BtDevice (p. 59), **e_bt_present_device** (p. 120)

9.18.2.8 char e_bt_list_local_paired_device (void)

Make a list of the bluetooth address of paired device.

Returns:

The number of device found

See also:

e_bt_local_paired_device (p. 120)

9.18.2.9 char e_bt_read_local_name (char * *name*)

Read the name of this e-puck's bluetooth module.

Parameters:

name A pointer to store the name

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.10 char e_bt_read_local_pin_number (char * *PIN*)

Read the PIN number of this e-puck's bluetooth module.

Parameters:

PIN A pointer to store the PIN number

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.11 char e_bt_release_SPP_link (void)

Unconnect from the current bluetooth device.

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.12 char e_bt_remove_local_paired_device (int *j*)

Remove a paired bluetooth device.

Parameters:

j The number of the paired device to remove from the e_bt_local_paired_device array.

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.13 int e_bt_reset (void)

Reset the bluetooth module.

Returns:

The version number

9.18.2.14 char e_bt_send_SPP_data (char * *data*, char *datalength*)

Send data to the current bluetooth device.

Warning:

send maximum 127 bytes if you are in non transparent mode

Parameters:

data The datas to send
datalength The length of the datas to send

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.15 char e_bt_tranparent_mode (void)

Change to transparent mode.

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.16 char e_bt_write_local_name (char * *name*)

Write the name on this e-puck's bluetooth module.

Parameters:

name A pointer to store the name

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.17 char e_bt_write_local_pin_number (char * *PIN*)

Write the PIN number on this e-puck's bluetooth module.

Parameters:

PIN A pointer to store the PIN number

Returns:

bluetooth error if one occur, 0 otherwise

9.18.3 Variable Documentation

9.18.3.1 unsigned char e_bt_local_paired_device[6 ∗8]

9.18.3.2 struct BtDevice e_bt_present_device[10]

An extern array containing all the bluetooth device detected. It's carried out by the fonction e_bt_find_-epuck

See also:

e_bt_find_epuck (p. 117)

9.18.3.3 struct BtEPuck e_bt_present_epuck[10]

An extern array containing all the e-puck detected. It's carried out by the fonction e_bt_find_epuck

See also:

e_bt_find_epuck (p. 117)

9.18.3.4 char local_bt_PIN[4]

9.19 bluetooth/e_bluetooth.h File Reference

Manage Bluetooth.

Data Structures

- struct **BtDevice**

  general struct for bluetooth device

- struct **BtEPuck**

  general struct for other e-puck

Functions

- int **e_bt_find_epuck** (void)

  Try to find other e-puck.

- char **e_bt_connect_epuck** (void)
- char **e_bt_read_local_pin_number** (char ∗PIN)

  Read the PIN number of this e-puck's bluetooth module.

- char **e_bt_read_local_name** (char ∗name)

  Read the name of this e-puck's bluetooth module.

- char **e_bt_write_local_pin_number** (char ∗PIN)

  Write the PIN number on this e-puck's bluetooth module.

- char **e_bt_write_local_name** (char ∗name)

  Write the name on this e-puck's bluetooth module.

- char **e_bt_etablish_SPP_link** (char ∗**address**)

  Try to connect to another bluetooth device.

- char **e_bt_release_SPP_link** (void)

  Unconnect from the current bluetooth device.

- char **e_bt_send_SPP_data** (char ∗**data**, char datalenght)

  Send data to the current bluetooth device.

- char **e_bt_tranparent_mode** (void)

  Change to transparent mode.

- void **e_bt_exit_tranparent_mode** (void)

  Exit from the transparent mode.

- char **e_bt_list_local_paired_device** (void)

  Make a list of the bluetooth address of paired device.

- char **e_bt_remove_local_paired_device** (int)
  Remove a paired bluetooth device.
- int **e_bt_inquiry** (struct **BtDevice** *device)
  Research all the accessible bluetooth devices.
- char **e_bt_get_friendly_name** (struct **BtDevice** *device)
  To get the friendly name of a bluetooth device.
- int **e_bt_reset** (void)
  Reset the bluetooth module.
- char **e_bt_factory_reset** (void)
  Factory reset of the bluetooth module.

Variables

- unsigned char **e_bt_local_paired_device** [6 *8]
- struct **BtDevice e_bt_present_device** [10]
- struct **BtEPuck e_bt_present_epuck** [10]

9.19.1 Detailed Description

Manage Bluetooth.

This module manage the bluetooth device.

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.19.2 Function Documentation

9.19.2.1 char e_bt_connect_epuck (void)

9.19.2.2 char e_bt_etablish_SPP_link (char * *address*)

Try to connect to another bluetooth device.

Parameters:

address A pointer on the device address which you want to connect

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.3 void e_bt_exit_tranparent_mode (void)

Exit from the transparent mode.

9.19.2.4 char e_bt_factory_reset (void)

Factory reset of the bluetooth module.

Warning:

use this function only if you are sure, your e-puck must be restarted, and renamed!!!

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.5 int e_bt_find_epuck (void)

Try to find other e-puck.

This function make global inguiry and check which device are e-puck, and list them in globales tables.

Returns:

number of e-puck found

See also:

e_bt_present_device (p. 120), **e_bt_present_epuck** (p. 120)

9.19.2.6 char e_bt_get_friendly_name (struct BtDevice * *device*)

To get the friendly name of a bluetooth device.

Parameters:

device A pointer on the device that you want the name

Returns:

bluetooth error if one occur, 0 otherwise

See also:

BtDevice (p. 59)

9.19.2.7 int e_bt_inquiry (struct BtDevice * *device*)

Research all the accessible bluetooth devices.

Parameters:

device A pointer to **BtDevice** (p. 59) to store all the caracteristics of each devices found

Returns:

the number of device found

See also:

BtDevice (p. 59), **e_bt_present_device** (p. 120)

9.19.2.8 char e_bt_list_local_paired_device (void)

Make a list of the bluetooth address of paired device.

Returns:

The number of device found

See also:

e_bt_local_paired_device (p. 120)

9.19.2.9 char e_bt_read_local_name (char * *name*)

Read the name of this e-puck's bluetooth module.

Parameters:

name A pointer to store the name

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.10 char e_bt_read_local_pin_number (char * *PIN*)

Read the PIN number of this e-puck's bluetooth module.

Parameters:

PIN A pointer to store the PIN number

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.11 char e_bt_release_SPP_link (void)

Unconnect from the current bluetooth device.

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.12 char e_bt_remove_local_paired_device (int *j*)

Remove a paired bluetooth device.

Parameters:

j The number of the paired device to remove from the e_bt_local_paired_device array.

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.13 int e_bt_reset (void)

Reset the bluetooth module.

Returns:

The version number

9.19.2.14 char e_bt_send_SPP_data (char * *data*, char *datalength*)

Send data to the current bluetooth device.

Warning:

send maximum 127 bytes if you are in non transparent mode

Parameters:

data The datas to send
datalength The length of the datas to send

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.15 char e_bt_tranparent_mode (void)

Change to transparent mode.

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.16 char e_bt_write_local_name (char * *name*)

Write the name on this e-puck's bluetooth module.

Parameters:

name A pointer to store the name

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.17 char e_bt_write_local_pin_number (char * *PIN*)

Write the PIN number on this e-puck's bluetooth module.

Parameters:

PIN A pointer to store the PIN number

Returns:

bluetooth error if one occur, 0 otherwise

9.19.3 Variable Documentation

9.19.3.1 unsigned char e_bt_local_paired_device[6 ∗8]

9.19.3.2 struct BtDevice e_bt_present_device[10]

An extern array containing all the bluetooth device detected. It's carried out by the fonction e_bt_find_-epuck

See also:

e_bt_find_epuck (p. 117)

9.19.3.3 struct BtEPuck e_bt_present_epuck[10]

An extern array containing all the e-puck detected. It's carried out by the fonction e_bt_find_epuck

See also:

e_bt_find_epuck (p. 117)

9.20 camera/fast_2_timer/e_calc.c File Reference

Calculate the timing for the camera (two timers).

```
#include "e_po3030k.h"
#include "../../motor_led/e_epuck_ports.h"
#include "../../I2C/e_I2C_protocol.h"
#include "../../motor_led/e_init_port.h"
```

Defines

- #define **ARRAY_ORIGINE_X** 210
- #define **ARRAY_ORIGINE_Y** 7
- #define **IRQ_PIX_LAT** 1

Functions

- int **e_po3030k_config_cam** (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_-width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)
- int **e_po3030k_get_bytes_per_pixel** (int color_mode)
- void **e_po3030k_init_cam** (void)

9.20.1 Detailed Description

Calculate the timing for the camera (two timers).

Author:

Philippe Retornaz

9.20.2 Define Documentation

9.20.2.1 #define ARRAY_ORIGINE_X 210

9.20.2.2 #define ARRAY_ORIGINE_Y 7

9.20.2.3 #define IRQ_PIX_LAT 1

9.20.3 Function Documentation

9.20.3.1 int e_po3030k_config_cam (unsigned int *sensor_x1*, unsigned int *sensor_y1*, unsigned int *sensor_width*, unsigned int *sensor_height*, unsigned int *zoom_fact_width*, unsigned int *zoom_fact_height*, int *color_mode*)

This function setup the internal timing of the camera to match the zoom, color mode and interest area.

Warning:

If the common deniminator of both zoom factor is 4 or 2, part of the subsampling is done by the camera (QQVGA = 4, QVGA = 2). This increase the framerate by respectivly 4 or 2. Moreover greyscale is twice faster than color mode.

Parameters:

sensor_x1 The X coordinate of the window's corner

sensor_y1 The Y coordinate of the window's corner

sensor_width the Width of the interest area, in FULL sampling scale

sensor_height The Height of the insterest area, in FULL sampling scale

zoom_fact_width The subsampling to apply for the window's Width

zoom_fact_height The subsampling to apply for the window's Height

color_mode The color mode in which the camera should be configured

Returns:

Zero if the settings are correct, non-zero if an error occur

See also:

e_po3030k_write_cam_registers (p. 187)

9.20.3.2 int e_po3030k_get_bytes_per_pixel (int *color_mode*)

Return the number of bytes per pixel in the given color mode

Parameters:

color_mode The given color mode

Returns:

The number of bytes per pixel in the given color mode

9.20.3.3 void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function

9.21 camera/slow_3_timer/e_calc.c File Reference

Calculate the timing for the camera (three timers).

```
#include "e_po3030k.h"
#include "./../../motor_LED/e_epuck_ports.h"
#include "./../../I2C/e_I2C_protocol.h"
#include "./../../motor_LED/e_init_port.h"
```

Defines

- #define **ARRAY_ORIGINE_X** 210
- #define **ARRAY_ORIGINE_Y** 7

Functions

- int **e_po3030k_config_cam** (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_-width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)
- int **e_po3030k_get_bytes_per_pixel** (int color_mode)
- void **e_po3030k_init_cam** (void)

9.21.1 Detailed Description

Calculate the timing for the camera (three timers).

9.21.2 Define Documentation

9.21.2.1 #define ARRAY_ORIGINE_X 210

9.21.2.2 #define ARRAY_ORIGINE_Y 7

9.21.3 Function Documentation

9.21.3.1 int e_po3030k_config_cam (unsigned int *sensor_x1*, unsigned int *sensor_y1*, unsigned int *sensor_width*, unsigned int *sensor_height*, unsigned int *zoom_fact_width*, unsigned int *zoom_fact_height*, int *color_mode*)

This function setup the internal timing of the camera to match the zoom, color mode and interest area.

Warning:

If the common deniminator of both zoom factor is 4 or 2, part of the subsampling is done by the camera (QQVGA = 4, QVGA = 2). This increase the framerate by respectivly 4 or 2.

Parameters:

sensor_x1 The X coordinate of the window's corner

sensor_y1 The Y coordinate of the window's corner

sensor_width the Width of the interest area, in FULL sampling scale

sensor_height The Height of the insterest area, in FULL sampling scale

zoom_fact_width The subsampling to apply for the window's Width

zoom_fact_height The subsampling to apply for the window's Height

color_mode The color mode in which the camera should be configured

Returns:

Zero if the settings are correct, non-zero if an error occur

See also:

e_po3030k_write_cam_registers (p. 187)

9.21.3.2 int e_po3030k_get_bytes_per_pixel (int *color_mode*)

Return the number of bytes per pixel in the given color mode

Parameters:

color_mode The given color mode

Returns:

The number of bytes per pixel in the given color mode

9.21.3.3 void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function

9.22 camera/fast_2_timer/e_po3030k.h File Reference

PO3030k library header (two timers).

Defines

- #define **PO3030K_FULL** 1
- #define **ARRAY_WIDTH** 640
- #define **ARRAY_HEIGHT** 480
- #define **GREY_SCALE_MODE** 0
- #define **RGB_565_MODE** 1
- #define **YUV_MODE** 2
- #define **MODE_VGA** 0x44
- #define **MODE_QVGA** 0x11
- #define **MODE_QQVGA** 0x33
- #define **SPEED_2** 0x00
- #define **SPEED_2_3** 0x10
- #define **SPEED_4** 0x20
- #define **SPEED_8** 0x30
- #define **SPEED_16** 0x40
- #define **SPEED_32** 0x50
- #define **SPEED_64** 0x60
- #define **SPEED_128** 0x70

Functions

- int **e_po3030k_config_cam** (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_-width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)
- int **e_po3030k_get_bytes_per_pixel** (int color_mode)
- void **e_po3030k_init_cam** (void)
- void **e_po3030k_write_cam_registers** (void)
- void **e_po3030k_launch_capture** (char ∗buf)
- int **e_po3030k_apply_timer_config** (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)
- int **e_po3030k_is_img_ready** (void)
- int **e_po3030k_set_color_mode** (int mode)
- int **e_po3030k_set_sampling_mode** (int mode)
- int **e_po3030k_set_speed** (int mode)
- int **e_po3030k_set_wx** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_wy** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_vsync** (unsigned int start, unsigned int stop, unsigned int col)
- void **e_po3030k_read_cam_registers** (void)
- int **e_po3030k_set_register** (unsigned char adr, unsigned char value)
- int **e_po3030k_get_register** (unsigned char adr, unsigned char ∗value)
- void **e_po3030k_set_bias** (unsigned char pixbias, unsigned char opbias)
- void **e_po3030k_set_integr_time** (unsigned long time)
- void **e_po3030k_set_mirror** (int vertical, int horizontal)
- void **e_po3030k_set_adc_offset** (unsigned char offset)
- void **e_po3030k_set_sepia** (int status)

- void **e_po3030k_set_lens_gain** (unsigned char red, unsigned char green, unsigned char blue)
- void **e_po3030k_set_edge_prop** (unsigned char gain, unsigned char tresh)
- void **e_po3030k_set_gamma_coef** (unsigned char array[12], char color)
- void **e_po3030k_write_gamma_coef** (void)
- int **e_po3030k_sync_register_array** (unsigned char start, unsigned char stop)
- void **e_po3030k_set_color_matrix** (unsigned char array[3 ∗3])
- void **e_po3030k_set_cb_cr_gain** (unsigned char cg11c, unsigned char cg22c)
- void **e_po3030k_set_brigh_contr** (unsigned char bright, unsigned char contrast)
- void **e_po3030k_set_sepia_tone** (unsigned char cb, unsigned char cr)
- void **e_po3030k_set_ww** (unsigned char ww)
- void **e_po3030k_set_awb_ae_tol** (unsigned char awbm, unsigned char aem)
- void **e_po3030k_set_ae_speed** (unsigned char b, unsigned char d)
- void **e_po3030k_set_exposure** (long t)
- void **e_po3030k_set_ref_exposure** (unsigned char exp)
- void **e_po3030k_set_max_min_exp** (unsigned int min, unsigned int max)
- void **e_po3030k_set_max_min_awb** (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratior, unsigned char ratiob)
- int **e_po3030k_set_weight_win** (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
- void **e_po3030k_set_awb_ae** (int awb, int ae)
- int **e_po3030k_set_color_gain** (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
- void **e_po3030k_set_flicker_mode** (int manual)
- void **e_po3030k_set_flicker_detection** (int hz50, int hz60)
- int **e_po3030k_set_flicker_man_set** (int hz50, int hz60, int fdm, int fk, int tol)

9.22.1 Detailed Description

PO3030k library header (two timers).

Author:

Philippe Retornaz

9.22.2 Define Documentation

9.22.2.1 #define ARRAY_HEIGHT 480

9.22.2.2 #define ARRAY_WIDTH 640

9.22.2.3 #define GREY_SCALE_MODE 0

9.22.2.4 #define MODE_QQVGA 0x33

9.22.2.5 #define MODE_QVGA 0x11

9.22.2.6 #define MODE_VGA 0x44

9.22.2.7 #define PO3030K_FULL 1

If you set this at 0, you save about 168 bytes of memory But you loose all advanced camera functions

9.22.2.8 #define RGB_565_MODE 1

9.22.2.9 #define SPEED_128 0x70

9.22.2.10 #define SPEED_16 0x40

9.22.2.11 #define SPEED_2 0x00

9.22.2.12 #define SPEED_2_3 0x10

9.22.2.13 #define SPEED_32 0x50

9.22.2.14 #define SPEED_4 0x20

9.22.2.15 #define SPEED_64 0x60

9.22.2.16 #define SPEED_8 0x30

9.22.2.17 #define YUV_MODE 2

9.22.3 Function Documentation

9.22.3.1 int e_po3030k_apply_timer_config (int *pixel_row*, int *pixel_col*, int *bpp*, int *pbp*, int *bbl*)

Modify the interrupt configuration

Warning:

This is an internal function, use *e_po3030k_config_cam*

Parameters:

pixel_row The number of row to take
pixel_col The number of pixel to take each *pixel_row*
bpp The number of byte per pixel
pbp The number of pixel to ignore between each pixel
bbl The number of row to ignore between each line

Returns:

Zero if OK, non-zero if the mode exceed internal data representation

See also:

e_po3030k_get_bytes_per_pixel (p. 145) and **e_po3030k_config_cam** (p. 144)

Modify the interrupt configuration

Warning:

This is an internal function, use *e_po3030k_config_cam*

Parameters:

pixel_row The number of row to take

pixel_col The number of pixel to take each *pixel_row*

bpp The number of byte per pixel

pbp The number of pixel to ignore between each pixel

bbl The number of row to ignore between each line

See also:

e_po3030k_get_bytes_per_pixel (p. 145) and **e_po3030k_config_cam** (p. 144)

9.22.3.2 int e_po3030k_config_cam (unsigned int *sensor_x1*, unsigned int *sensor_y1*, unsigned int *sensor_width*, unsigned int *sensor_height*, unsigned int *zoom_fact_width*, unsigned int *zoom_fact_height*, int *color_mode*)

This function setup the internal timing of the camera to match the zoom, color mode and interest area.

Warning:

If the common deniminator of both zoom factor is 4 or 2, part of the subsampling is done by the camera (QQVGA = 4, QVGA = 2). This increase the framerate by respectivly 4 or 2. Moreover greyscale is twice faster than color mode.

Parameters:

sensor_x1 The X coordinate of the window's corner

sensor_y1 The Y coordinate of the window's corner

sensor_width the Width of the interest area, in FULL sampling scale

sensor_height The Height of the insterest area, in FULL sampling scale

zoom_fact_width The subsampling to apply for the window's Width

zoom_fact_height The subsampling to apply for the window's Height

color_mode The color mode in which the camera should be configured

Returns:

Zero if the settings are correct, non-zero if an error occur

See also:

e_po3030k_write_cam_registers (p. 187)

This function setup the internal timing of the camera to match the zoom, color mode and interest area.

Warning:

If the common deniminator of both zoom factor is 4 or 2, part of the subsampling is done by the camera (QQVGA = 4, QVGA = 2). This increase the framerate by respectivly 4 or 2.

Parameters:

sensor_x1 The X coordinate of the window's corner

sensor_y1 The Y coordinate of the window's corner

sensor_width the Width of the interest area, in FULL sampling scale

sensor_height The Height of the insterest area, in FULL sampling scale

zoom_fact_width The subsampling to apply for the window's Width

zoom_fact_height The subsampling to apply for the window's Height

color_mode The color mode in which the camera should be configured

Returns:

Zero if the settings are correct, non-zero if an error occur

See also:

e_po3030k_write_cam_registers (p. 187)

9.22.3.3 int e_po3030k_get_bytes_per_pixel (int *color_mode*)

Return the number of bytes per pixel in the given color mode

Parameters:

color_mode The given color mode

Returns:

The number of bytes per pixel in the given color mode

9.22.3.4 int e_po3030k_get_register (unsigned char *adr*, unsigned char * *value*)

Get the register *adr* value

Parameters:

adr The address

value The pointer to the value to write to

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_set_register (p. 183)

9.22.3.5 void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function

9.22.3.6 int e_po3030k_is_img_ready (void)

Check if the current capture is finished

Returns:

Zero if the current capture is in progress, non-zero if the capture is done.

See also:

e_po3030k_launch_capture (p. 193)

9.22.3.7 void e_po3030k_launch_capture (char * *buf*)

Launch a capture in the *buf* buffer

Parameters:

buf The buffer to write to

See also:

e_po3030k_config_cam (p. 144) and **e_po3030k_is_img_ready** (p. 193)

9.22.3.8 void e_po3030k_read_cam_registers (void)

Read the camera register

See also:

e_po3030k_write_cam_registers (p. 187)

9.22.3.9 void e_po3030k_set_adc_offset (unsigned char *offset*)

Set the Analog to Digital Converter offset

Parameters:

offset The offset

See also:

Datasheet p.28

9.22.3.10 void e_po3030k_set_ae_speed (unsigned char *b*, unsigned char *d*)

Set AE speed

Parameters:

b AE speed factor when exposure time is decreasing (4 lower bits only)
d AE speed factor when exposure time is increasing (4 lower bits only)

See also:

Datasheet p.44

9.22.3.11 void e_po3030k_set_awb_ae (int *awb*, int *ae*)

Enable/Disable AWB and AE

Parameters:

awb 1 mean AWB enabled, 0 mean disabled

ae 1 mean AE enabled, 0 mean disabled

See also:

Datasheet p. 50

9.22.3.12 void e_po3030k_set_awb_ae_tol (unsigned char *awbm*, unsigned char *aem*)

Set AWB/AE tolerence margin

Parameters:

awbm AWV Margin (4 lower bits only)

aem AW Margin (4 lower bits only)

See also:

Datasheet p.44

9.22.3.13 void e_po3030k_set_bias (unsigned char *pixbias*, unsigned char *opbias*)

Set the Pixel and amplificator bias Increasing the bias produce better image quality, but increase camera's power consumption

Parameters:

pixbias The pixel bias

opbias The Amplificator bias

See also:

Datasheet p.22

9.22.3.14 void e_po3030k_set_brigh_contr (unsigned char *bright*, unsigned char *contrast*)

Set the Brightness & Contrast

Parameters:

bright The Brightness (signed 1+7bits fixed point format)

contrast The Contrast

See also:

Datasheet p. 40

9.22.3.15 void e_po3030k_set_cb_cr_gain (unsigned char *cg11c*, unsigned char *cg22c*)

Set The color gain (Cb/Cr)

Parameters:

cg11c Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])

cg22c Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:

Datasheet p. 40

9.22.3.16 int e_po3030k_set_color_gain (unsigned char *global*, unsigned char *red*, unsigned char *green1*, unsigned char *green2*, unsigned char *blue*)

Set the gains of the camera

Parameters:

global The global gain $\in \{0, 79\}$

red The red pixel's gain (fixed point [2:6] format)

green1 The green pixel near read one gain ([2:6] format)

green2 The green pixel near blue one gain ([2:6] format)

blue The blue pixel's gain ([2:6] format)

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.23-24

9.22.3.17 void e_po3030k_set_color_matrix (unsigned char *array*[3 *∗*3])

9.22.3.18 int e_po3030k_set_color_mode (int *mode*)

Set the camera color mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The color mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 31, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.22.3.19 void e_po3030k_set_edge_prop (unsigned char *gain*, unsigned char *tresh*)

Set Edge properties

Parameters:

gain Edge gain & moire factor (fixed point [2:3] format)

tresh Edge Enhancement threshold

See also:

Datasheet p.36

9.22.3.20 void e_po3030k_set_exposure (long *t*)

Set exposure time

Parameters:

t Exposure time, LSB is in 1/64 line time

Warning:

Only writable if AE is disabled

See also:

Datasheet p.45

9.22.3.21 void e_po3030k_set_flicker_detection (int *hz50*, int *hz60*)

Set the 50/60Hz flicker detection

Parameters:

hz50 Non-zero mean 50Hz flicker detection enabled (default disabled)

hz60 Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

If Automatic mode is enabled and both 50Hz and 60Hz are disabled, camera will enable both. By default, the camera automatically detect 50 and 60Hz flicker.

See also:

Datasheet p.29 **e_po3030k_set_flicker_mode()** (p. 134) **e_po3030k_set_flicker_man_set()** (p. 133)

9.22.3.22 int e_po3030k_set_flicker_man_set (int *hz50*, int *hz60*, int *fdm*, int *fk*, int *tol*)

Set the camera's manual flicker's detection setting

Parameters:

hz50 The Hz for the 50Hz detection

hz60 The Hz for the 60Hz detection

fdm Flicker duration mode

fk Flicker count step

tol Flicker tolerance

Warning:

You must have set the mode (image size, color) before calling this function

Returns:

Non-zero if an error occur, 0 if OK

See also:

Datasheet p.29-30 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_mode()** (p. 134)

9.22.3.23 void e_po3030k_set_flicker_mode (int *manual*)

Set flicker detection mode

Parameters:

manual Non-zero mean manual mode is enabled (default automatic mode enabled)

See also:

Datasheet p.29 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_man_set()** (p. 133)

9.22.3.24 void e_po3030k_set_gamma_coef (unsigned char *array*[12], char *color*)

Set gamma coefficient

Warning:

This feature need extra care from the user

Parameters:

array Gamma coefficient array

color First two bytes : - 0b01 => Green

- 0b00 => Red
- else => Blue

See also:

Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef

9.22.3.25 void e_po3030k_set_integr_time (unsigned long *time*)

Set the pixel intergration time This is counted in line-time interval. See dataseet p.25 for more information

Parameters:

time The integration time (fixed point [14:6] format)

See also:

Datasheet p.25

9.22.3.26 void e_po3030k_set_lens_gain (unsigned char *red*, unsigned char *green*, unsigned char *blue*)

Set lens shading gain

Parameters:

red Lens gain for red pixel $\in \{0, 15\}$

green Lens gain for green pixel $\in \{0, 15\}$

blue Lens gain for blue pixel $\in \{0, 15\}$

See also:

Datasheet p.36

9.22.3.27 void e_po3030k_set_max_min_awb (unsigned char *minb*, unsigned char *maxb*, unsigned char *minr*, unsigned char *maxr*, unsigned char *ratior*, unsigned char *ratiob*)

Set the minimum and maximum red and blue gain in AWB mode

Parameters:

minb The minimum blue gain

maxb The maximum blue gain

minr The minimum red gain

maxr The maximum red gain

ratior The red gain ratio

ratiob The blue gain ratio

See also:

Datasheet p. 47-48

9.22.3.28 void e_po3030k_set_max_min_exp (unsigned int *max*, unsigned int *min*)

Set the minimum and maximum exposure time in AE mode

Parameters:

min The minimum exposure time

max The maximum exposure time

See also:

Datasheet p.46-47

9.22.3.29 void e_po3030k_set_mirror (int *vertical*, int *horizontal*)

Enable/Disable horizontal or vertical mirror

Parameters:

vertical Set to 1 when vertical mirror is enabled, 0 if disabled

horizontal Set to 1 when horizontal mirror is enabled, 0 if disabled

See also:

Datasheet p.27

9.22.3.30 void e_po3030k_set_ref_exposure (unsigned char *exp*)

Set the reference exposure. The average brightness which the AE should have

Parameters:

exp The target exposure level

See also:

Datasheet p.45

9.22.3.31 int e_po3030k_set_register (unsigned char *adr*, unsigned char *value*)

Set the register *adr* to value *value*

Parameters:

adr The address

value The value

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_get_register (p. 176)

9.22.3.32 int e_po3030k_set_sampling_mode (int *mode*)

Set the camera sampling mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given sampling mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 28 and **e_po3030k_config_cam** (p. 144)

9.22.3.33 void e_po3030k_set_sepia (int *status*)

Enable/Disable Sepia color

Parameters:

status Set *status* to 1 to enable, 0 to disable

See also:

Datasheet p.34 and 74

9.22.3.34 void e_po3030k_set_sepia_tone (unsigned char *cb*, unsigned char *cr*)

Set The color tone at sepia color condition

Parameters:

cb Cb tone

cr Cr tone

See also:

e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74

9.22.3.35 int e_po3030k_set_speed (int *mode*)

Set the camera speed

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given speed

Returns:

Zero if OK, non-zero if unknow mode

See also:

Datasheet p. 26 and **e_po3030k_config_cam** (p. 144)

9.22.3.36 int e_po3030k_set_vsync (unsigned int *start*, unsigned int *stop*, unsigned int *col*)

Set the camera window VSYNC coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

col The start/stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.42-43, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.22.3.37 int e_po3030k_set_weight_win (unsigned int *x1*, unsigned int *x2*, unsigned int *y1*, unsigned int *y2*)

Set the Weighting Window coordinate

Parameters:

x1 The X1 coordinate $\in \{211, x2\}$

x2 The X2 coordinate $\in \{x1 + 1, 423\}$

y1 The Y1 coordinate $\in \{160, y2\}$

y2 The Y2 coordinate $\in \{y1 + 1, 319\}$

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 49

9.22.3.38 void e_po3030k_set_ww (unsigned char *ww*)

Set the Center weight (Back Light compensation) Control parameter

Parameters:

ww Center weight (4 lower bits only)

See also:

Datasheet p.44

9.22.3.39 int e_po3030k_set_wx (unsigned int *start*, unsigned int *stop*)

Set the camera window X coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start column

stop The stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, **e_po3030k_write_cam_registers** (p. 187), **e_po3030k_set_wy** (p. 186) and **e_po3030k_config_cam** (p. 144)

9.22.3.40 int e_po3030k_set_wy (unsigned int *start*, unsigned int *stop*)

Set the camera window Y coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

9.22.3.41 int e_po3030k_sync_register_array (unsigned char *start*, unsigned char *stop*)

Write every known register between address start and stop (inclusivly).

Warning:

It's better to set the configuration with appropriate functions and then write all registers with e_po3030k_WriteCamRegisters

Parameters:

start The beginning address of the write

stop The last write address

Returns:

The number of register written

See also:

e_po3030k_write_cam_registers (p. 187)

9.22.3.42 void e_po3030k_write_cam_registers (void)

The Po3030k module keep in memory the state of each register the camera has. When you configure the camera, it only alter the internal register state, not the camera one. This function write the internal register state in the camera.

See also:

e_po3030k_read_cam_registers (p. 177)

9.22.3.43 void e_po3030k_write_gamma_coef (void)

This special function write directly the Gamma coefficient and Gamma color select into camera register.

Warning:

This function need extra care from the user

See also:

e_po3030k_set_gamma_coef (p. 181)

9.23 camera/slow_3_timer/e_po3030k.h File Reference

PO3030k library header (three timers).

Defines

- #define **PO3030K_FULL** 1
- #define **ARRAY_WIDTH** 640
- #define **ARRAY_HEIGHT** 480
- #define **GREY_SCALE_MODE** 0
- #define **RGB_565_MODE** 1
- #define **YUV_MODE** 2
- #define **MODE_VGA** 0x44
- #define **MODE_QVGA** 0x11
- #define **MODE_QQVGA** 0x33
- #define **SPEED_2** 0x00
- #define **SPEED_2_3** 0x10
- #define **SPEED_4** 0x20
- #define **SPEED_8** 0x30
- #define **SPEED_16** 0x40
- #define **SPEED_32** 0x50
- #define **SPEED_64** 0x60
- #define **SPEED_128** 0x70

Functions

- int **e_po3030k_config_cam** (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_-width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)
- int **e_po3030k_get_bytes_per_pixel** (int color_mode)
- void **e_po3030k_init_cam** (void)
- void **e_po3030k_write_cam_registers** (void)
- void **e_po3030k_launch_capture** (char ∗buf)
- void **e_po3030k_apply_timer_config** (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)
- int **e_po3030k_is_img_ready** (void)
- int **e_po3030k_set_color_mode** (int mode)
- int **e_po3030k_set_sampling_mode** (int mode)
- int **e_po3030k_set_speed** (int mode)
- int **e_po3030k_set_wx** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_wy** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_vsync** (unsigned int start, unsigned int stop, unsigned int col)
- void **e_po3030k_read_cam_registers** (void)
- int **e_po3030k_set_register** (unsigned char adr, unsigned char value)
- int **e_po3030k_get_register** (unsigned char adr, unsigned char ∗value)
- void **e_po3030k_set_bias** (unsigned char pixbias, unsigned char opbias)
- void **e_po3030k_set_integr_time** (unsigned long time)
- void **e_po3030k_set_mirror** (int vertical, int horizontal)
- void **e_po3030k_set_adc_offset** (unsigned char offset)
- void **e_po3030k_set_sepia** (int status)

- void **e_po3030k_set_lens_gain** (unsigned char red, unsigned char green, unsigned char blue)
- void **e_po3030k_set_edge_prop** (unsigned char gain, unsigned char tresh)
- void **e_po3030k_set_gamma_coef** (unsigned char array[12], char color)
- void **e_po3030k_write_gamma_coef** (void)
- int **e_po3030k_sync_register_array** (unsigned char start, unsigned char stop)
- void **e_po3030k_set_color_matrix** (unsigned char array[3 ∗3])
- void **e_po3030k_set_cb_cr_gain** (unsigned char cg11c, unsigned char cg22c)
- void **e_po3030k_set_brigh_contr** (unsigned char bright, unsigned char contrast)
- void **e_po3030k_set_sepia_tone** (unsigned char cb, unsigned char cr)
- void **e_po3030k_set_ww** (unsigned char ww)
- void **e_po3030k_set_awb_ae_tol** (unsigned char awbm, unsigned char aem)
- void **e_po3030k_set_ae_speed** (unsigned char b, unsigned char d)
- void **e_po3030k_set_exposure** (long t)
- void **e_po3030k_set_ref_exposure** (unsigned char exp)
- void **e_po3030k_set_max_min_exp** (unsigned int min, unsigned int max)
- void **e_po3030k_set_max_min_awb** (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratior, unsigned char ratiob)
- int **e_po3030k_set_weight_win** (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
- void **e_po3030k_set_awb_ae** (int awb, int ae)
- int **e_po3030k_set_color_gain** (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
- void **e_po3030k_set_flicker_mode** (int manual)
- void **e_po3030k_set_flicker_detection** (int hz50, int hz60)
- int **e_po3030k_set_flicker_man_set** (int hz50, int hz60, int fdm, int fk, int tol)

9.23.1 Detailed Description

PO3030k library header (three timers).

Author:

Philippe Retornaz

9.23.2 Define Documentation

9.23.2.1 #define ARRAY_HEIGHT 480

9.23.2.2 #define ARRAY_WIDTH 640

9.23.2.3 #define GREY_SCALE_MODE 0

9.23.2.4 #define MODE_QQVGA 0x33

9.23.2.5 #define MODE_QVGA 0x11

9.23.2.6 #define MODE_VGA 0x44

9.23.2.7 #define PO3030K_FULL 1

If you set this at 0, you save about 168 bytes of memory But you loose all advanced camera functions

9.23.2.8 #define RGB_565_MODE 1

9.23.2.9 #define SPEED_128 0x70

9.23.2.10 #define SPEED_16 0x40

9.23.2.11 #define SPEED_2 0x00

9.23.2.12 #define SPEED_2_3 0x10

9.23.2.13 #define SPEED_32 0x50

9.23.2.14 #define SPEED_4 0x20

9.23.2.15 #define SPEED_64 0x60

9.23.2.16 #define SPEED_8 0x30

9.23.2.17 #define YUV_MODE 2

9.23.3 Function Documentation

9.23.3.1 void e_po3030k_apply_timer_config (int *pixel_row*, int *pixel_col*, int *bpp*, int *pbp*, int *bbl*)

Modify the interrupt configuration

Warning:

This is an internal function, use *e_po3030k_config_cam*

Parameters:

pixel_row The number of row to take
pixel_col The number of pixel to take each *pixel_row*
bpp The number of byte per pixel
pbp The number of pixel to ignore between each pixel
bbl The number of row to ignore between each line

Returns:

Zero if OK, non-zero if the mode exceed internal data representation

See also:

e_po3030k_get_bytes_per_pixel (p. 145) and **e_po3030k_config_cam** (p. 144)

Modify the interrupt configuration

Warning:

This is an internal function, use *e_po3030k_config_cam*

Parameters:

pixel_row The number of row to take

pixel_col The number of pixel to take each *pixel_row*

bpp The number of byte per pixel

pbp The number of pixel to ignore between each pixel

bbl The number of row to ignore between each line

See also:

e_po3030k_get_bytes_per_pixel (p. 145) and **e_po3030k_config_cam** (p. 144)

9.23.3.2 int e_po3030k_config_cam (unsigned int *sensor_x1*, unsigned int *sensor_y1*, unsigned int *sensor_width*, unsigned int *sensor_height*, unsigned int *zoom_fact_width*, unsigned int *zoom_fact_height*, int *color_mode*)

This function setup the internal timing of the camera to match the zoom, color mode and interest area.

Warning:

If the common deniminator of both zoom factor is 4 or 2, part of the subsampling is done by the camera (QQVGA = 4, QVGA = 2). This increase the framerate by respectivly 4 or 2. Moreover greyscale is twice faster than color mode.

Parameters:

sensor_x1 The X coordinate of the window's corner

sensor_y1 The Y coordinate of the window's corner

sensor_width the Width of the interest area, in FULL sampling scale

sensor_height The Height of the insterest area, in FULL sampling scale

zoom_fact_width The subsampling to apply for the window's Width

zoom_fact_height The subsampling to apply for the window's Height

color_mode The color mode in which the camera should be configured

Returns:

Zero if the settings are correct, non-zero if an error occur

See also:

e_po3030k_write_cam_registers (p. 187)

This function setup the internal timing of the camera to match the zoom, color mode and interest area.

Warning:

If the common deniminator of both zoom factor is 4 or 2, part of the subsampling is done by the camera (QQVGA = 4, QVGA = 2). This increase the framerate by respectivly 4 or 2.

Parameters:

sensor_x1 The X coordinate of the window's corner

sensor_y1 The Y coordinate of the window's corner

sensor_width the Width of the interest area, in FULL sampling scale

sensor_height The Height of the insterest area, in FULL sampling scale

zoom_fact_width The subsampling to apply for the window's Width

zoom_fact_height The subsampling to apply for the window's Height

color_mode The color mode in which the camera should be configured

Returns:

Zero if the settings are correct, non-zero if an error occur

See also:

e_po3030k_write_cam_registers (p. 187)

9.23.3.3 int e_po3030k_get_bytes_per_pixel (int *color_mode*)

Return the number of bytes per pixel in the given color mode

Parameters:

color_mode The given color mode

Returns:

The number of bytes per pixel in the given color mode

9.23.3.4 int e_po3030k_get_register (unsigned char *adr*, unsigned char * *value*)

Get the register *adr* value

Parameters:

adr The address

value The pointer to the value to write to

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_set_register (p. 183)

9.23.3.5 void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function

9.23.3.6 int e_po3030k_is_img_ready (void)

Check if the current capture is finished

Returns:

Zero if the current capture is in progress, non-zero if the capture is done.

See also:

e_po3030k_launch_capture (p. 193)

9.23.3.7 void e_po3030k_launch_capture (char * *buf*)

Launch a capture in the *buf* buffer

Parameters:

buf The buffer to write to

See also:

e_po3030k_config_cam (p. 144) and **e_po3030k_is_img_ready** (p. 193)

9.23.3.8 void e_po3030k_read_cam_registers (void)

Read the camera register

See also:

e_po3030k_write_cam_registers (p. 187)

9.23.3.9 void e_po3030k_set_adc_offset (unsigned char *offset*)

Set the Analog to Digital Converter offset

Parameters:

offset The offset

See also:

Datasheet p.28

9.23.3.10 void e_po3030k_set_ae_speed (unsigned char *b*, unsigned char *d*)

Set AE speed

Parameters:

b AE speed factor when exposure time is decreasing (4 lower bits only)

d AE speed factor when exposure time is increasing (4 lower bits only)

See also:

Datasheet p.44

9.23.3.11 void e_po3030k_set_awb_ae (int *awb*, int *ae*)

Enable/Disable AWB and AE

Parameters:

awb 1 mean AWB enabled, 0 mean disabled

ae 1 mean AE enabled, 0 mean disabled

See also:

Datasheet p. 50

9.23.3.12 void e_po3030k_set_awb_ae_tol (unsigned char *awbm*, unsigned char *aem*)

Set AWB/AE tolerence margin

Parameters:

awbm AWV Margin (4 lower bits only)

aem AW Margin (4 lower bits only)

See also:

Datasheet p.44

9.23.3.13 void e_po3030k_set_bias (unsigned char *pixbias*, unsigned char *opbias*)

Set the Pixel and amplificator bias Increasing the bias produce better image quality, but increase camera's power consumption

Parameters:

pixbias The pixel bias

opbias The Amplificator bias

See also:

Datasheet p.22

9.23.3.14 void e_po3030k_set_brigh_contr (unsigned char *bright*, unsigned char *contrast*)

Set the Brightness & Contrast

Parameters:

bright The Brightness (signed 1+7bits fixed point format)

contrast The Contrast

See also:

Datasheet p. 40

9.23.3.15 void e_po3030k_set_cb_cr_gain (unsigned char *cg11c*, unsigned char *cg22c*)

Set The color gain (Cb/Cr)

Parameters:

cg11c Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])

cg22c Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:

Datasheet p. 40

9.23.3.16 int e_po3030k_set_color_gain (unsigned char *global*, unsigned char *red*, unsigned char *green1*, unsigned char *green2*, unsigned char *blue*)

Set the gains of the camera

Parameters:

global The global gain $\in \{0, 79\}$

red The red pixel's gain (fixed point [2:6] format)

green1 The green pixel near read one gain ([2:6] format)

green2 The green pixel near blue one gain ([2:6] format)

blue The blue pixel's gain ([2:6] format)

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.23-24

9.23.3.17 void e_po3030k_set_color_matrix (unsigned char *array*[3 *∗*3])

9.23.3.18 int e_po3030k_set_color_mode (int *mode*)

Set the camera color mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The color mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 31, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.23.3.19 void e_po3030k_set_edge_prop (unsigned char *gain*, unsigned char *tresh*)

Set Edge properties

Parameters:

gain Edge gain & moire factor (fixed point [2:3] format)

tresh Edge Enhancement threshold

See also:

Datasheet p.36

9.23.3.20 void e_po3030k_set_exposure (long *t*)

Set exposure time

Parameters:

t Exposure time, LSB is in 1/64 line time

Warning:

Only writable if AE is disabled

See also:

Datasheet p.45

9.23.3.21 void e_po3030k_set_flicker_detection (int *hz50*, int *hz60*)

Set the 50/60Hz flicker detection

Parameters:

hz50 Non-zero mean 50Hz flicker detection enabled (default disabled)

hz60 Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

If Automatic mode is enabled and both 50Hz and 60Hz are disabled, camera will enable both. By default, the camera automatically detect 50 and 60Hz flicker.

See also:

Datasheet p.29 **e_po3030k_set_flicker_mode()** (p. 134) **e_po3030k_set_flicker_man_set()** (p. 133)

9.23.3.22 int e_po3030k_set_flicker_man_set (int *hz50*, int *hz60*, int *fdm*, int *fk*, int *tol*)

Set the camera's manual flicker's detection setting

Parameters:

hz50 The Hz for the 50Hz detection

hz60 The Hz for the 60Hz detection

fdm Flicker duration mode

fk Flicker count step

tol Flicker tolerance

Warning:

You must have set the mode (image size, color) before calling this function

Returns:

Non-zero if an error occur, 0 if OK

See also:

Datasheet p.29-30 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_mode()** (p. 134)

9.23.3.23 void e_po3030k_set_flicker_mode (int *manual*)

Set flicker detection mode

Parameters:

manual Non-zero mean manual mode is enabled (default automatic mode enabled)

See also:

Datasheet p.29 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_man_set()** (p. 133)

9.23.3.24 void e_po3030k_set_gamma_coef (unsigned char *array*[12], char *color*)

Set gamma coefficient

Warning:

This feature need extra care from the user

Parameters:

array Gamma coefficient array

color First two bytes : - 0b01 => Green

- 0b00 => Red
- else => Blue

See also:

Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef

9.23.3.25 void e_po3030k_set_integr_time (unsigned long *time*)

Set the pixel intergration time This is counted in line-time interval. See dataseet p.25 for more information

Parameters:

time The integration time (fixed point [14:6] format)

See also:

Datasheet p.25

9.23.3.26 void e_po3030k_set_lens_gain (unsigned char *red*, unsigned char *green*, unsigned char *blue*)

Set lens shading gain

Parameters:

red Lens gain for red pixel $\in \{0, 15\}$

green Lens gain for green pixel $\in \{0, 15\}$

blue Lens gain for blue pixel $\in \{0, 15\}$

See also:

Datasheet p.36

9.23.3.27 void e_po3030k_set_max_min_awb (unsigned char *minb*, unsigned char *maxb*, unsigned char *minr*, unsigned char *maxr*, unsigned char *ratior*, unsigned char *ratiob*)

Set the minimum and maximum red and blue gain in AWB mode

Parameters:

minb The minimum blue gain

maxb The maximum blue gain

minr The minimum red gain

maxr The maximum red gain

ratior The red gain ratio

ratiob The blue gain ratio

See also:

Datasheet p. 47-48

9.23.3.28 void e_po3030k_set_max_min_exp (unsigned int *max*, unsigned int *min*)

Set the minimum and maximum exposure time in AE mode

Parameters:

min The minimum exposure time

max The maximum exposure time

See also:

Datasheet p.46-47

9.23.3.29 void e_po3030k_set_mirror (int *vertical*, int *horizontal*)

Enable/Disable horizontal or vertical mirror

Parameters:

vertical Set to 1 when vertical mirror is enabled, 0 if disabled

horizontal Set to 1 when horizontal mirror is enabled, 0 if disabled

See also:

Datasheet p.27

9.23.3.30 void e_po3030k_set_ref_exposure (unsigned char *exp*)

Set the reference exposure. The average brightness which the AE should have

Parameters:

exp The target exposure level

See also:

Datasheet p.45

9.23.3.31 int e_po3030k_set_register (unsigned char *adr*, unsigned char *value*)

Set the register *adr* to value *value*

Parameters:

adr The address

value The value

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_get_register (p. 176)

9.23.3.32 int e_po3030k_set_sampling_mode (int *mode*)

Set the camera sampling mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given sampling mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 28 and **e_po3030k_config_cam** (p. 144)

9.23.3.33 void e_po3030k_set_sepia (int *status*)

Enable/Disable Sepia color

Parameters:

status Set *status* to 1 to enable, 0 to disable

See also:

Datasheet p.34 and 74

9.23.3.34 void e_po3030k_set_sepia_tone (unsigned char *cb*, unsigned char *cr*)

Set The color tone at sepia color condition

Parameters:

cb Cb tone

cr Cr tone

See also:

e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74

9.23.3.35 int e_po3030k_set_speed (int *mode*)

Set the camera speed

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given speed

Returns:

Zero if OK, non-zero if unknow mode

See also:

Datasheet p. 26 and **e_po3030k_config_cam** (p. 144)

9.23.3.36 int e_po3030k_set_vsync (unsigned int *start*, unsigned int *stop*, unsigned int *col*)

Set the camera window VSYNC coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

col The start/stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.42-43, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.23.3.37 int e_po3030k_set_weight_win (unsigned int *x1*, unsigned int *x2*, unsigned int *y1*, unsigned int *y2*)

Set the Weighting Window coordinate

Parameters:

x1 The X1 coordinate $\in \{211, x2\}$

x2 The X2 coordinate $\in \{x1 + 1, 423\}$

y1 The Y1 coordinate $\in \{160, y2\}$

y2 The Y2 coordinate $\in \{y1 + 1, 319\}$

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 49

9.23.3.38 void e_po3030k_set_ww (unsigned char *ww*)

Set the Center weight (Back Light compensation) Control parameter

Parameters:

ww Center weight (4 lower bits only)

See also:

Datasheet p.44

9.23.3.39 int e_po3030k_set_wx (unsigned int *start*, unsigned int *stop*)

Set the camera window X coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start column

stop The stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, **e_po3030k_write_cam_registers** (p. 187), **e_po3030k_set_wy** (p. 186) and **e_po3030k_config_cam** (p. 144)

9.23.3.40 int e_po3030k_set_wy (unsigned int *start*, unsigned int *stop*)

Set the camera window Y coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

9.23.3.41 int e_po3030k_sync_register_array (unsigned char *start*, unsigned char *stop*)

Write every known register between address start and stop (inclusivly).

Warning:

It's better to set the configuration with appropriate functions and then write all registers with e_po3030k_WriteCamRegisters

Parameters:

start The beginning address of the write

stop The last write address

Returns:

The number of register written

See also:

e_po3030k_write_cam_registers (p. 187)

9.23.3.42 void e_po3030k_write_cam_registers (void)

The Po3030k module keep in memory the state of each register the camera has. When you configure the camera, it only alter the internal register state, not the camera one. This function write the internal register state in the camera.

See also:

e_po3030k_read_cam_registers (p. 177)

9.23.3.43 void e_po3030k_write_gamma_coef (void)

This special function write directly the Gamma coefficient and Gamma color select into camera register.

Warning:

This function need extra care from the user

See also:

e_po3030k_set_gamma_coef (p. 181)

9.24 camera/fast_2_timer/e_registers.c File Reference

Manage po3030k registers (two timers).

```
#include "../../I2C/e_I2C_protocol.h"
#include "e_po3030k.h"
```

Defines

- #define **MCLK** ((long) 14745600)
- #define **MCLK_P_NS** 0.067816840278
- #define **ARRAY_ORIGINE_X** 210
- #define **ARRAY_ORIGINE_Y** 7
- #define **BASE_D1** 0
- #define **BASE_D2** 0
- #define **BASE_D3** 0
- #define **BASE_D4** 0
- #define **DEVICE_ID** 0xDC
- #define **FRAME_WIDTH** 0x0353
- #define **FRAME_HEIGTH** 0x01e8
- #define **WINDOW_X1_BASE** 9
- #define **WINDOW_Y1_BASE** 13
- #define **WINDOW_X2_BASE** 17
- #define **WINDOW_Y2_BASE** 21
- #define **BIAS_BASE** 25
- #define **COLGAIN_BASE** 29
- #define **INTEGR_BASE** 39
- #define **SPEED_ADDR** (45 - BASE_D1)
- #define **MIRROR_BASE** 47
- #define **SAMPLING_ADDR** (51 - BASE_D2)
- #define **ADCOFF_BASE** (53 - BASE_D2)
- #define **FLICKM_BASE** (55 - BASE_D2)
- #define **FLICKP_BASE** 59
- #define **COLOR_M_ADDR** (67 - BASE_D3)
- #define **MODE_R5G6B5** 0x08
- #define **MODE_YUV** 0x02
- #define **MODE_GRAYSCALE** 0x0c
- #define **SEPIA_BASE** 69
- #define **LENSG_BASE** 73
- #define **EDGE_BASE** 79
- #define **GAMMA_BASE** 83
- #define **COLOR_COEF_BASE** 107
- #define **CBCRGAIN_BASE** 125
- #define **BRICTR_BASE** 129
- #define **SEPIATONE_BASE** 133
- #define **VSYNCSTART_BASE** (145 - BASE_D4)
- #define **VSYNCSTOP_BASE** (149 - BASE_D4)
- #define **VSYNCCOL_BASE** (153 - BASE_D4)
- #define **WW_BASE** 157
- #define **AWVAETOL_BASE** 159

- #define **AESPEED_BASE** 161
- #define **EXPOSURE_BASE** 163
- #define **REFREPO_BASE** 169
- #define **MINMAXEXP_BASE** 175
- #define **MINMAXAWB_BASE** 183
- #define **WEIGHWIN_BASE** 195
- #define **AWBAEENABLE_BASE** 211
- #define **GAMMASELCOL_BASE** 215
- #define **NB_REGISTERS** (sizeof(**cam_reg**)/(2∗sizeof(**cam_reg**[0])))

Functions

- void **e_po3030k_write_cam_registers** (void)
- void **e_po3030k_read_cam_registers** (void)
- int **e_po3030k_set_color_mode** (int mode)
- int **e_po3030k_set_sampling_mode** (int mode)
- int **e_po3030k_set_speed** (int mode)
- int **e_po3030k_set_wx** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_wy** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_vsync** (unsigned int start, unsigned int stop, unsigned int col)
- int **e_po3030k_set_register** (unsigned char adr, unsigned char value)
- int **e_po3030k_get_register** (unsigned char adr, unsigned char ∗value)
- void **e_po3030k_set_bias** (unsigned char pixbias, unsigned char opbias)
- int **e_po3030k_set_color_gain** (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
- void **e_po3030k_set_integr_time** (unsigned long time)
- void **e_po3030k_set_mirror** (int vertical, int horizontal)
- void **e_po3030k_set_adc_offset** (unsigned char offset)
- void **e_po3030k_set_sepia** (int status)
- void **e_po3030k_set_lens_gain** (unsigned char red, unsigned char green, unsigned char blue)
- void **e_po3030k_set_edge_prop** (unsigned char gain, unsigned char tresh)
- void **e_po3030k_set_gamma_coef** (unsigned char array[12], char color)
- void **e_po3030k_write_gamma_coef** (void)
- int **e_po3030k_sync_register_array** (unsigned char start, unsigned char stop)
- void **e_po3030k_SetColorMatrix** (unsigned char array[3 ∗3])
- void **e_po3030k_set_cb_cr_gain** (unsigned char cg11c, unsigned char cg22c)
- void **e_po3030k_set_brigh_contr** (unsigned char bright, unsigned char contrast)
- void **e_po3030k_set_sepia_tone** (unsigned char cb, unsigned char cr)
- void **e_po3030k_set_ww** (unsigned char ww)
- void **e_po3030k_set_awb_ae_tol** (unsigned char awbm, unsigned char aem)
- void **e_po3030k_set_ae_speed** (unsigned char b, unsigned char d)
- void **e_po3030k_set_exposure** (long t)
- void **e_po3030k_set_ref_exposure** (unsigned char exp)
- void **e_po3030k_set_max_min_exp** (unsigned int max, unsigned int min)
- void **e_po3030k_set_max_min_awb** (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratior, unsigned char ratiob)
- int **e_po3030k_set_weight_win** (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
- void **e_po3030k_set_awb_ae** (int awb, int ae)
- void **e_po3030k_set_flicker_mode** (int manual)
- void **e_po3030k_set_flicker_detection** (int hz50, int hz60)
- static long **po3030k_get_pixelclock** (void)
- int **e_po3030k_set_flicker_man_set** (int hz50, int hz60, int fdm, int fk, int tol)

Variables

- static unsigned char **cam_reg** [ ]

9.24.1 Detailed Description

Manage po3030k registers (two timers).

Author:

Philippe Retornaz

9.24.2 Define Documentation

9.24.2.1 #define ADCOFF_BASE (53 - BASE_D2)

9.24.2.2 #define AESPEED_BASE 161

9.24.2.3 #define ARRAY_ORIGINE_X 210

9.24.2.4 #define ARRAY_ORIGINE_Y 7

9.24.2.5 #define AWBAEENABLE_BASE 211

9.24.2.6 #define AWVAETOL_BASE 159

9.24.2.7 #define BASE_D1 0

9.24.2.8 #define BASE_D2 0

9.24.2.9 #define BASE_D3 0

9.24.2.10 #define BASE_D4 0

9.24.2.11 #define BIAS_BASE 25

9.24.2.12 #define BRICTR_BASE 129

9.24.2.13 #define CBCRGAIN_BASE 125

9.24.2.14 #define COLGAIN_BASE 29

9.24.2.15 #define COLOR_COEF_BASE 107

9.24.2.16 #define COLOR_M_ADDR (67 - BASE_D3)

9.24.2.17 #define DEVICE_ID 0xDC

9.24.2.18 #define EDGE_BASE 79

9.24.2.19 #define EXPOSURE_BASE 163

9.24.2.20 #define FLICKM_BASE (55 - BASE_D2)

9.24.2.21 #define FLICKP_BASE 59

9.24.2.22 #define FRAME_HEIGTH 0x01e8

9.24.2.23 #define FRAME_WIDTH 0x0353

9.24.2.24 #define GAMMA_BASE 83

9.24.2.25 #define GAMMASELCOL_BASE 215

9.24.2.26 #define INTEGR_BASE 39

9.24.2.27 #define LENSG_BASE 73

9.24.2.28 #define MCLK ((long) 14745600)

9.24.2.29 #define MCLK_P_NS 0.067816840278

9.24.2.30 #define MINMAXAWB_BASE 183

Parameters:

adr The address

value The pointer to the value to write to

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_set_register (p. 183)

9.24.3.2 void e_po3030k_read_cam_registers (void)

Read the camera register

See also:

e_po3030k_write_cam_registers (p. 187)

9.24.3.3 void e_po3030k_set_adc_offset (unsigned char *offset*)

Set the Analog to Digital Converter offset

Parameters:

offset The offset

See also:

Datasheet p.28

9.24.3.4 void e_po3030k_set_ae_speed (unsigned char *b*, unsigned char *d*)

Set AE speed

Parameters:

b AE speed factor when exposure time is decreasing (4 lower bits only)

d AE speed factor when exposure time is increasing (4 lower bits only)

See also:

Datasheet p.44

9.24.3.5 void e_po3030k_set_awb_ae (int *awb*, int *ae*)

Enable/Disable AWB and AE

Parameters:

awb 1 mean AWB enabled, 0 mean disabled

ae 1 mean AE enabled, 0 mean disabled

See also:

Datasheet p. 50

9.24.3.6 void e_po3030k_set_awb_ae_tol (unsigned char *awbm*, unsigned char *aem*)

Set AWB/AE tolerence margin

Parameters:

awbm AWV Margin (4 lower bits only)

aem AW Margin (4 lower bits only)

See also:

Datasheet p.44

9.24.3.7 void e_po3030k_set_bias (unsigned char *pixbias*, unsigned char *opbias*)

Set the Pixel and amplificator bias Increasing the bias produce better image quality, but increase camera's power consumption

Parameters:

pixbias The pixel bias

opbias The Amplificator bias

See also:

Datasheet p.22

9.24.3.8 void e_po3030k_set_brigh_contr (unsigned char *bright*, unsigned char *contrast*)

Set the Brightness & Contrast

Parameters:

bright The Brightness (signed 1+7bits fixed point format)

contrast The Contrast

See also:

Datasheet p. 40

9.24.3.9 void e_po3030k_set_cb_cr_gain (unsigned char *cg11c*, unsigned char *cg22c*)

Set The color gain (Cb/Cr)

Parameters:

cg11c Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])

cg22c Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:

Datasheet p. 40

9.24.3.10 int e_po3030k_set_color_gain (unsigned char *global*, unsigned char *red*, unsigned char *green1*, unsigned char *green2*, unsigned char *blue*)

Set the gains of the camera

Parameters:

global The global gain $\in \{0, 79\}$

red The red pixel's gain (fixed point [2:6] format)

green1 The green pixel near read one gain ([2:6] format)

green2 The green pixel near blue one gain ([2:6] format)

blue The blue pixel's gain ([2:6] format)

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.23-24

9.24.3.11 int e_po3030k_set_color_mode (int *mode*)

Set the camera color mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The color mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 31, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.24.3.12 void e_po3030k_set_edge_prop (unsigned char *gain*, unsigned char *tresh*)

Set Edge properties

Parameters:

gain Edge gain & moire factor (fixed point [2:3] format)

tresh Edge Enhancement threshold

See also:

Datasheet p.36

9.24.3.13 void e_po3030k_set_exposure (long *t*)

Set exposure time

Parameters:

t Exposure time, LSB is in 1/64 line time

Warning:

Only writable if AE is disabled

See also:

Datasheet p.45

9.24.3.14 void e_po3030k_set_flicker_detection (int *hz50*, int *hz60*)

Set the 50/60Hz flicker detection

Parameters:

hz50 Non-zero mean 50Hz flicker detection enabled (default disabled)

hz60 Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

If Automatic mode is enabled and both 50Hz and 60Hz are disabled, camera will enable both. By default, the camera automatically detect 50 and 60Hz flicker.

See also:

Datasheet p.29 **e_po3030k_set_flicker_mode()** (p. 134) **e_po3030k_set_flicker_man_set()** (p. 133)

9.24.3.15 int e_po3030k_set_flicker_man_set (int *hz50*, int *hz60*, int *fdm*, int *fk*, int *tol*)

Set the camera's manual flicker's detection setting

Parameters:

hz50 The Hz for the 50Hz detection

hz60 The Hz for the 60Hz detection

fdm Flicker duration mode

fk Flicker count step

tol Flicker tolerance

Warning:

You must have set the mode (image size, color) before calling this function

Returns:

Non-zero if an error occur, 0 if OK

See also:

Datasheet p.29-30 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_mode()** (p. 134)

9.24.3.16 void e_po3030k_set_flicker_mode (int *manual*)

Set flicker detection mode

Parameters:

manual Non-zero mean manual mode is enabled (default automatic mode enabled)

See also:

Datasheet p.29 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_man_set()** (p. 133)

9.24.3.17 void e_po3030k_set_gamma_coef (unsigned char *array*[12], char *color*)

Set gamma coefficient

Warning:

This feature need extra care from the user

Parameters:

array Gamma coefficient array

color First two bytes : - 0b01 => Green

- 0b00 => Red
- else => Blue

See also:

Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef

9.24.3.18 void e_po3030k_set_integr_time (unsigned long *time*)

Set the pixel intergration time This is counted in line-time interval. See dataseet p.25 for more information

Parameters:

time The integration time (fixed point [14:6] format)

See also:

Datasheet p.25

9.24.3.19 void e_po3030k_set_lens_gain (unsigned char *red*, unsigned char *green*, unsigned char *blue*)

Set lens shading gain

Parameters:

red Lens gain for red pixel $\in \{0, 15\}$

green Lens gain for green pixel $\in \{0, 15\}$

blue Lens gain for blue pixel $\in \{0, 15\}$

See also:

Datasheet p.36

9.24.3.20 void e_po3030k_set_max_min_awb (unsigned char *minb*, unsigned char *maxb*, unsigned char *minr*, unsigned char *maxr*, unsigned char *ratior*, unsigned char *ratiob*)

Set the minimum and maximum red and blue gain in AWB mode

Parameters:

minb The minimum blue gain

maxb The maximum blue gain

minr The minimum red gain

maxr The maximum red gain

ratior The red gain ratio

ratiob The blue gain ratio

See also:

Datasheet p. 47-48

9.24.3.21 void e_po3030k_set_max_min_exp (unsigned int *max*, unsigned int *min*)

Set the minimum and maximum exposure time in AE mode

Parameters:

min The minimum exposure time

max The maximum exposure time

See also:

Datasheet p.46-47

9.24.3.22 void e_po3030k_set_mirror (int *vertical*, int *horizontal*)

Enable/Disable horizontal or vertical mirror

Parameters:

vertical Set to 1 when vertical mirror is enabled, 0 if disabled

horizontal Set to 1 when horizontal mirror is enabled, 0 if disabled

See also:

Datasheet p.27

9.24.3.23 void e_po3030k_set_ref_exposure (unsigned char *exp*)

Set the reference exposure. The average brightness which the AE should have

Parameters:

exp The target exposure level

See also:

Datasheet p.45

9.24.3.24 int e_po3030k_set_register (unsigned char *adr*, unsigned char *value*)

Set the register *adr* to value *value*

Parameters:

adr The address

value The value

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_get_register (p. 176)

9.24.3.25 int e_po3030k_set_sampling_mode (int *mode*)

Set the camera sampling mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given sampling mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 28 and **e_po3030k_config_cam** (p. 144)

9.24.3.26 void e_po3030k_set_sepia (int *status*)

Enable/Disable Sepia color

Parameters:

status Set *status* to 1 to enable, 0 to disable

See also:

Datasheet p.34 and 74

9.24.3.27 void e_po3030k_set_sepia_tone (unsigned char *cb*, unsigned char *cr*)

Set The color tone at sepia color condition

Parameters:

cb Cb tone

cr Cr tone

See also:

e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74

9.24.3.28 int e_po3030k_set_speed (int *mode*)

Set the camera speed

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given speed

Returns:

Zero if OK, non-zero if unknow mode

See also:

Datasheet p. 26 and **e_po3030k_config_cam** (p. 144)

9.24.3.29 int e_po3030k_set_vsync (unsigned int *start*, unsigned int *stop*, unsigned int *col*)

Set the camera window VSYNC coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

col The start/stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.42-43, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.24.3.30 int e_po3030k_set_weight_win (unsigned int *x1*, unsigned int *x2*, unsigned int *y1*, unsigned int *y2*)

Set the Weighting Window coordinate

Parameters:

x1 The X1 coordinate $\in \{211, x2\}$

x2 The X2 coordinate $\in \{x1 + 1, 423\}$

y1 The Y1 coordinate $\in \{160, y2\}$

y2 The Y2 coordinate $\in \{y1 + 1, 319\}$

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 49

9.24.3.31 void e_po3030k_set_ww (unsigned char *ww*)

Set the Center weight (Back Light compensation) Control parameter

Parameters:

ww Center weight (4 lower bits only)

See also:

Datasheet p.44

9.24.3.32 int e_po3030k_set_wx (unsigned int *start*, unsigned int *stop*)

Set the camera window X coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start column

stop The stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, **e_po3030k_write_cam_registers** (p. 187), **e_po3030k_set_wy** (p. 186) and **e_po3030k_config_cam** (p. 144)

9.24.3.33 int e_po3030k_set_wy (unsigned int *start*, unsigned int *stop*)

Set the camera window Y coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

9.24.3.34 void e_po3030k_SetColorMatrix (unsigned char *array*[3 ∗3])

Set color correction coefficient

Parameters:

array Color coefficient matrix (3x3), sign[7] | integer [6:5] | fractional [4:0]

See also:

Datasheet p. 39

9.24.3.35 int e_po3030k_sync_register_array (unsigned char *start*, unsigned char *stop*)

Write every known register between address start and stop (inclusivly).

Warning:

It's better to set the configuration with appropriate functions and then write all registers with e_-po3030k_WriteCamRegisters

Parameters:

start The beginning address of the write

stop The last write address

Returns:

The number of register written

See also:

e_po3030k_write_cam_registers (p. 187)

9.24.3.36 void e_po3030k_write_cam_registers (void)

The Po3030k module keep in memory the state of each register the camera has. When you configure the camera, it only alter the internal register state, not the camera one. This function write the internal register state in the camera.

See also:

e_po3030k_read_cam_registers (p. 177)

9.24.3.37 void e_po3030k_write_gamma_coef (void)

This special function write directly the Gamma coefficient and Gamma color select into camera register.

Warning:

This function need extra care from the user

See also:

e_po3030k_set_gamma_coef (p. 181)

9.24.3.38 static long po3030k_get_pixelclock (void) [static]

9.24.4 Variable Documentation

9.24.4.1 unsigned char cam_reg[ ] [static]

9.25 camera/slow_3_timer/e_registers.c File Reference

Manage po3030k registers (three timers).

```
#include "./../../I2C/e_I2C_protocol.h"
#include "e_po3030k.h"
```

Defines

- #define **MCLK** ((long) 14745600)
- #define **MCLK_P_NS** 0.067816840278
- #define **ARRAY_ORIGINE_X** 210
- #define **ARRAY_ORIGINE_Y** 7
- #define **BASE_D1** 0
- #define **BASE_D2** 0
- #define **BASE_D3** 0
- #define **BASE_D4** 0
- #define **DEVICE_ID** 0xDC
- #define **FRAME_WIDTH** 0x0353
- #define **FRAME_HEIGTH** 0x01e8
- #define **WINDOW_X1_BASE** 9
- #define **WINDOW_Y1_BASE** 13
- #define **WINDOW_X2_BASE** 17
- #define **WINDOW_Y2_BASE** 21
- #define **BIAS_BASE** 25
- #define **COLGAIN_BASE** 29
- #define **INTEGR_BASE** 39
- #define **SPEED_ADDR** (45 - BASE_D1)
- #define **MIRROR_BASE** 47
- #define **SAMPLING_ADDR** (51 - BASE_D2)
- #define **ADCOFF_BASE** (53 - BASE_D2)
- #define **FLICKM_BASE** (55 - BASE_D2)
- #define **FLICKP_BASE** 59
- #define **COLOR_M_ADDR** (67 - BASE_D3)
- #define **MODE_R5G6B5** 0x08
- #define **MODE_YUV** 0x02
- #define **MODE_GRAYSCALE** 0x0c
- #define **SEPIA_BASE** 69
- #define **LENSG_BASE** 73
- #define **EDGE_BASE** 79
- #define **GAMMA_BASE** 83
- #define **COLOR_COEF_BASE** 107
- #define **CBCRGAIN_BASE** 125
- #define **BRICTR_BASE** 129
- #define **SEPIATONE_BASE** 133
- #define **VSYNCSTART_BASE** (145 - BASE_D4)
- #define **VSYNCSTOP_BASE** (149 - BASE_D4)
- #define **VSYNCCOL_BASE** (153 - BASE_D4)
- #define **WW_BASE** 157
- #define **AWVAETOL_BASE** 159

- #define **AESPEED_BASE** 161
- #define **EXPOSURE_BASE** 163
- #define **REFREPO_BASE** 169
- #define **MINMAXEXP_BASE** 175
- #define **MINMAXAWB_BASE** 183
- #define **WEIGHWIN_BASE** 195
- #define **AWBAEENABLE_BASE** 211
- #define **GAMMASELCOL_BASE** 215
- #define **NB_REGISTERS** (sizeof(**cam_reg**)/(2∗sizeof(**cam_reg**[0])))

Functions

- void **e_po3030k_write_cam_registers** (void)
- void **e_po3030k_read_cam_registers** (void)
- int **e_po3030k_set_color_mode** (int mode)
- int **e_po3030k_set_sampling_mode** (int mode)
- int **e_po3030k_set_speed** (int mode)
- int **e_po3030k_set_wx** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_wy** (unsigned int start, unsigned int stop)
- int **e_po3030k_set_vsync** (unsigned int start, unsigned int stop, unsigned int col)
- int **e_po3030k_set_register** (unsigned char adr, unsigned char value)
- int **e_po3030k_get_register** (unsigned char adr, unsigned char ∗value)
- void **e_po3030k_set_bias** (unsigned char pixbias, unsigned char opbias)
- int **e_po3030k_set_color_gain** (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
- void **e_po3030k_set_integr_time** (unsigned long time)
- void **e_po3030k_set_mirror** (int vertical, int horizontal)
- void **e_po3030k_set_adc_offset** (unsigned char offset)
- void **e_po3030k_set_sepia** (int status)
- void **e_po3030k_set_lens_gain** (unsigned char red, unsigned char green, unsigned char blue)
- void **e_po3030k_set_edge_prop** (unsigned char gain, unsigned char tresh)
- void **e_po3030k_set_gamma_coef** (unsigned char array[12], char color)
- void **e_po3030k_write_gamma_coef** (void)
- int **e_po3030k_sync_register_array** (unsigned char start, unsigned char stop)
- void **e_po3030k_SetColorMatrix** (unsigned char array[3 ∗3])
- void **e_po3030k_set_cb_cr_gain** (unsigned char cg11c, unsigned char cg22c)
- void **e_po3030k_set_brigh_contr** (unsigned char bright, unsigned char contrast)
- void **e_po3030k_set_sepia_tone** (unsigned char cb, unsigned char cr)
- void **e_po3030k_set_ww** (unsigned char ww)
- void **e_po3030k_set_awb_ae_tol** (unsigned char awbm, unsigned char aem)
- void **e_po3030k_set_ae_speed** (unsigned char b, unsigned char d)
- void **e_po3030k_set_exposure** (long t)
- void **e_po3030k_set_ref_exposure** (unsigned char exp)
- void **e_po3030k_set_max_min_exp** (unsigned int max, unsigned int min)
- void **e_po3030k_set_max_min_awb** (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratior, unsigned char ratiob)
- int **e_po3030k_set_weight_win** (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
- void **e_po3030k_set_awb_ae** (int awb, int ae)
- void **e_po3030k_set_flicker_mode** (int manual)
- void **e_po3030k_set_flicker_detection** (int hz50, int hz60)
- static long **po3030k_get_pixelclock** (void)
- int **e_po3030k_set_flicker_man_set** (int hz50, int hz60, int fdm, int fk, int tol)

Variables

- static unsigned char **cam_reg** [ ]

9.25.1 Detailed Description

Manage po3030k registers (three timers).

Author:

Philippe Retornaz

9.25.2 Define Documentation

9.25.2.1 #define ADCOFF_BASE (53 - BASE_D2)

9.25.2.2 #define AESPEED_BASE 161

9.25.2.3 #define ARRAY_ORIGINE_X 210

9.25.2.4 #define ARRAY_ORIGINE_Y 7

9.25.2.5 #define AWBAEENABLE_BASE 211

9.25.2.6 #define AWVAETOL_BASE 159

9.25.2.7 #define BASE_D1 0

9.25.2.8 #define BASE_D2 0

9.25.2.9 #define BASE_D3 0

9.25.2.10 #define BASE_D4 0

9.25.2.11 #define BIAS_BASE 25

9.25.2.12 #define BRICTR_BASE 129

9.25.2.13 #define CBCRGAIN_BASE 125

9.25.2.14 #define COLGAIN_BASE 29

9.25.2.15 #define COLOR_COEF_BASE 107

9.25.2.16 #define COLOR_M_ADDR (67 - BASE_D3)

9.25.2.17 #define DEVICE_ID 0xDC

9.25.2.18 #define EDGE_BASE 79

9.25.2.19 #define EXPOSURE_BASE 163

9.25.2.20 #define FLICKM_BASE (55 - BASE_D2)

9.25.2.21 #define FLICKP_BASE 59

9.25.2.22 #define FRAME_HEIGTH 0x01e8

9.25.2.23 #define FRAME_WIDTH 0x0353

9.25.2.24 #define GAMMA_BASE 83

9.25.2.25 #define GAMMASELCOL_BASE 215

9.25.2.26 #define INTEGR_BASE 39

9.25.2.27 #define LENSG_BASE 73

9.25.2.28 #define MCLK ((long) 14745600)

9.25.2.29 #define MCLK_P_NS 0.067816840278

9.25.2.30 #define MINMAXAWB_BASE 183

Parameters:

adr The address

value The pointer to the value to write to

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_set_register (p. 183)

9.25.3.2 void e_po3030k_read_cam_registers (void)

Read the camera register

See also:

e_po3030k_write_cam_registers (p. 187)

9.25.3.3 void e_po3030k_set_adc_offset (unsigned char *offset*)

Set the Analog to Digital Converter offset

Parameters:

offset The offset

See also:

Datasheet p.28

9.25.3.4 void e_po3030k_set_ae_speed (unsigned char *b*, unsigned char *d*)

Set AE speed

Parameters:

b AE speed factor when exposure time is decreasing (4 lower bits only)

d AE speed factor when exposure time is increasing (4 lower bits only)

See also:

Datasheet p.44

9.25.3.5 void e_po3030k_set_awb_ae (int *awb*, int *ae*)

Enable/Disable AWB and AE

Parameters:

awb 1 mean AWB enabled, 0 mean disabled

ae 1 mean AE enabled, 0 mean disabled

See also:

Datasheet p. 50

9.25.3.6 void e_po3030k_set_awb_ae_tol (unsigned char *awbm*, unsigned char *aem*)

Set AWB/AE tolerence margin

Parameters:

awbm AWV Margin (4 lower bits only)

aem AW Margin (4 lower bits only)

See also:

Datasheet p.44

9.25.3.7 void e_po3030k_set_bias (unsigned char *pixbias*, unsigned char *opbias*)

Set the Pixel and amplificator bias Increasing the bias produce better image quality, but increase camera's power consumption

Parameters:

pixbias The pixel bias

opbias The Amplificator bias

See also:

Datasheet p.22

9.25.3.8 void e_po3030k_set_brigh_contr (unsigned char *bright*, unsigned char *contrast*)

Set the Brightness & Contrast

Parameters:

bright The Brightness (signed 1+7bits fixed point format)

contrast The Contrast

See also:

Datasheet p. 40

9.25.3.9 void e_po3030k_set_cb_cr_gain (unsigned char *cg11c*, unsigned char *cg22c*)

Set The color gain (Cb/Cr)

Parameters:

cg11c Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])

cg22c Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:

Datasheet p. 40

9.25.3.10 int e_po3030k_set_color_gain (unsigned char *global*, unsigned char *red*, unsigned char *green1*, unsigned char *green2*, unsigned char *blue*)

Set the gains of the camera

Parameters:

global The global gain $\in \{0, 79\}$

red The red pixel's gain (fixed point [2:6] format)

green1 The green pixel near read one gain ([2:6] format)

green2 The green pixel near blue one gain ([2:6] format)

blue The blue pixel's gain ([2:6] format)

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.23-24

9.25.3.11 int e_po3030k_set_color_mode (int *mode*)

Set the camera color mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The color mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 31, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.25.3.12 void e_po3030k_set_edge_prop (unsigned char *gain*, unsigned char *tresh*)

Set Edge properties

Parameters:

gain Edge gain & moire factor (fixed point [2:3] format)

tresh Edge Enhancement threshold

See also:

Datasheet p.36

9.25.3.13 void e_po3030k_set_exposure (long *t*)

Set exposure time

Parameters:

t Exposure time, LSB is in 1/64 line time

Warning:

Only writable if AE is disabled

See also:

Datasheet p.45

9.25.3.14 void e_po3030k_set_flicker_detection (int *hz50*, int *hz60*)

Set the 50/60Hz flicker detection

Parameters:

hz50 Non-zero mean 50Hz flicker detection enabled (default disabled)

hz60 Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

If Automatic mode is enabled and both 50Hz and 60Hz are disabled, camera will enable both. By default, the camera automatically detect 50 and 60Hz flicker.

See also:

Datasheet p.29 **e_po3030k_set_flicker_mode()** (p. 134) **e_po3030k_set_flicker_man_set()** (p. 133)

9.25.3.15 int e_po3030k_set_flicker_man_set (int *hz50*, int *hz60*, int *fdm*, int *fk*, int *tol*)

Set the camera's manual flicker's detection setting

Parameters:

hz50 The Hz for the 50Hz detection

hz60 The Hz for the 60Hz detection

fdm Flicker duration mode

fk Flicker count step

tol Flicker tolerance

Warning:

You must have set the mode (image size, color) before calling this function

Returns:

Non-zero if an error occur, 0 if OK

See also:

Datasheet p.29-30 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_mode()** (p. 134)

9.25.3.16 void e_po3030k_set_flicker_mode (int *manual*)

Set flicker detection mode

Parameters:

manual Non-zero mean manual mode is enabled (default automatic mode enabled)

See also:

Datasheet p.29 **e_po3030k_set_flicker_detection()** (p. 133) **e_po3030k_set_flicker_man_set()** (p. 133)

9.25.3.17 void e_po3030k_set_gamma_coef (unsigned char *array*[12], char *color*)

Set gamma coefficient

Warning:

This feature need extra care from the user

Parameters:

array Gamma coefficient array

color First two bytes : - 0b01 => Green

- 0b00 => Red
- else => Blue

See also:

Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef

9.25.3.18 void e_po3030k_set_integr_time (unsigned long *time*)

Set the pixel intergration time This is counted in line-time interval. See dataseet p.25 for more information

Parameters:

time The integration time (fixed point [14:6] format)

See also:

Datasheet p.25

9.25.3.19 void e_po3030k_set_lens_gain (unsigned char *red*, unsigned char *green*, unsigned char *blue*)

Set lens shading gain

Parameters:

red Lens gain for red pixel $\in \{0, 15\}$

green Lens gain for green pixel $\in \{0, 15\}$

blue Lens gain for blue pixel $\in \{0, 15\}$

See also:

Datasheet p.36

9.25.3.20 void e_po3030k_set_max_min_awb (unsigned char *minb*, unsigned char *maxb*, unsigned char *minr*, unsigned char *maxr*, unsigned char *ratior*, unsigned char *ratiob*)

Set the minimum and maximum red and blue gain in AWB mode

Parameters:

minb The minimum blue gain

maxb The maximum blue gain

minr The minimum red gain

maxr The maximum red gain

ratior The red gain ratio

ratiob The blue gain ratio

See also:

Datasheet p. 47-48

9.25.3.21 void e_po3030k_set_max_min_exp (unsigned int *max*, unsigned int *min*)

Set the minimum and maximum exposure time in AE mode

Parameters:

min The minimum exposure time

max The maximum exposure time

See also:

Datasheet p.46-47

9.25.3.22 void e_po3030k_set_mirror (int *vertical*, int *horizontal*)

Enable/Disable horizontal or vertical mirror

Parameters:

vertical Set to 1 when vertical mirror is enabled, 0 if disabled

horizontal Set to 1 when horizontal mirror is enabled, 0 if disabled

See also:

Datasheet p.27

9.25.3.23 void e_po3030k_set_ref_exposure (unsigned char *exp*)

Set the reference exposure. The average brightness which the AE should have

Parameters:

exp The target exposure level

See also:

Datasheet p.45

9.25.3.24 int e_po3030k_set_register (unsigned char *adr*, unsigned char *value*)

Set the register *adr* to value *value*

Parameters:

adr The address

value The value

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavly add an internal function to register.c

See also:

e_po3030k_get_register (p. 176)

9.25.3.25 int e_po3030k_set_sampling_mode (int *mode*)

Set the camera sampling mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given sampling mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 28 and **e_po3030k_config_cam** (p. 144)

9.25.3.26 void e_po3030k_set_sepia (int *status*)

Enable/Disable Sepia color

Parameters:

status Set *status* to 1 to enable, 0 to disable

See also:

Datasheet p.34 and 74

9.25.3.27 void e_po3030k_set_sepia_tone (unsigned char *cb*, unsigned char *cr*)

Set The color tone at sepia color condition

Parameters:

cb Cb tone

cr Cr tone

See also:

e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74

9.25.3.28 int e_po3030k_set_speed (int *mode*)

Set the camera speed

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode The given speed

Returns:

Zero if OK, non-zero if unknow mode

See also:

Datasheet p. 26 and **e_po3030k_config_cam** (p. 144)

9.25.3.29 int e_po3030k_set_vsync (unsigned int *start*, unsigned int *stop*, unsigned int *col*)

Set the camera window VSYNC coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

col The start/stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.42-43, **e_po3030k_write_cam_registers** (p. 187) and **e_po3030k_config_cam** (p. 144)

9.25.3.30 int e_po3030k_set_weight_win (unsigned int *x1*, unsigned int *x2*, unsigned int *y1*, unsigned int *y2*)

Set the Weighting Window coordinate

Parameters:

x1 The X1 coordinate $\in \{211, x2\}$

x2 The X2 coordinate $\in \{x1 + 1, 423\}$

y1 The Y1 coordinate $\in \{160, y2\}$

y2 The Y2 coordinate $\in \{y1 + 1, 319\}$

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 49

9.25.3.31 void e_po3030k_set_ww (unsigned char *ww*)

Set the Center weight (Back Light compensation) Control parameter

Parameters:

ww Center weight (4 lower bits only)

See also:

Datasheet p.44

9.25.3.32 int e_po3030k_set_wx (unsigned int *start*, unsigned int *stop*)

Set the camera window X coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start column

stop The stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, **e_po3030k_write_cam_registers** (p. 187), **e_po3030k_set_wy** (p. 186) and **e_po3030k_config_cam** (p. 144)

9.25.3.33 int e_po3030k_set_wy (unsigned int *start*, unsigned int *stop*)

Set the camera window Y coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start The start row

stop The stop row

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

9.25.3.34 void e_po3030k_SetColorMatrix (unsigned char *array*[3 ∗3])

Set color correction coefficient

Parameters:

array Color coefficient matrix (3x3), sign[7] | integer [6:5] | fractional [4:0]

See also:

Datasheet p. 39

9.25.3.35 int e_po3030k_sync_register_array (unsigned char *start*, unsigned char *stop*)

Write every known register between address start and stop (inclusivly).

Warning:

It's better to set the configuration with appropriate functions and then write all registers with e_-po3030k_WriteCamRegisters

Parameters:

start The beginning address of the write

stop The last write address

Returns:

The number of register written

See also:

e_po3030k_write_cam_registers (p. 187)

9.25.3.36 void e_po3030k_write_cam_registers (void)

The Po3030k module keep in memory the state of each register the camera has. When you configure the camera, it only alter the internal register state, not the camera one. This function write the internal register state in the camera.

See also:

e_po3030k_read_cam_registers (p. 177)

9.25.3.37 void e_po3030k_write_gamma_coef (void)

This special function write directly the Gamma coefficient and Gamma color select into camera register.

Warning:

This function need extra care from the user

See also:

e_po3030k_set_gamma_coef (p. 181)

9.25.3.38 static long po3030k_get_pixelclock (void) [static]

9.25.4 Variable Documentation

9.25.4.1 unsigned char cam_reg[ ] [static]

9.26 camera/fast_2_timer/e_timers.c File Reference

Manage camera's interrupts (two timers).

```
#include <p30f6014A.h>
#include "../../motor_led/e_epuck_ports.h"
#include "e_po3030k.h"
```

Functions

- void **__attribute__** ((interrupt, auto_psv))

  The VSYNC interrupt. This interrupt is called every time the Vertical sync signal is asserted This mean that the picture is comming from the camera (we will have the first line soon).

- static void **init_timer5** (void)
- static void **init_timer4** (void)
- void **e_po3030k_launch_capture** (char ∗buf)
- int **e_po3030k_apply_timer_config** (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)
- int **e_po3030k_is_img_ready** (void)

Variables

- char ∗ **_po3030k_buffer**
- int **_po3030k_img_ready**
- static int **blank_row_betw**
- int **_po3030k_current_row**
- int **_po3030k_row**
- char **_po3030k_line_conf** [330]

9.26.1 Detailed Description

Manage camera's interrupts (two timers).

Author:

Philippe Retornaz

9.26.2 Function Documentation

9.26.2.1 void __attribute__ ((interrupt, auto_psv))

The VSYNC interrupt. This interrupt is called every time the Vertical sync signal is asserted This mean that the picture is comming from the camera (we will have the first line soon).

The Pixel Clock interrupt. This interrupt is called every time the Pixel clock signal is asserted This mean that the next byte is ready to be read.

9.26.2.2 int e_po3030k_apply_timer_config (int *pixel_row*, int *pixel_col*, int *bpp*, int *pbp*, int *bbl*)

Modify the interrupt configuration

Warning:

This is an internal function, use *e_po3030k_config_cam*

Parameters:

pixel_row The number of row to take

pixel_col The number of pixel to take each *pixel_row*

bpp The number of byte per pixel

pbp The number of pixel to ignore between each pixel

bbl The number of row to ignore between each line

Returns:

Zero if OK, non-zero if the mode exceed internal data representation

See also:

e_po3030k_get_bytes_per_pixel (p. 145) and **e_po3030k_config_cam** (p. 144)

9.26.2.3 int e_po3030k_is_img_ready (void)

Check if the current capture is finished

Returns:

Zero if the current capture is in progress, non-zero if the capture is done.

See also:

e_po3030k_launch_capture (p. 193)

9.26.2.4 void e_po3030k_launch_capture (char * *buf*)

Launch a capture in the *buf* buffer

Parameters:

buf The buffer to write to

See also:

e_po3030k_config_cam (p. 144) and **e_po3030k_is_img_ready** (p. 193)

9.26.2.5 static void init_timer4 (void) [static]

9.26.2.6 static void init_timer5 (void) [static]

9.26.3 Variable Documentation

9.26.3.1 char* _po3030k_buffer

The buffer to write to

9.26.3.2 int _po3030k_current_row

9.26.3.3 int _po3030k_img_ready

The flag to tell, the image is ready or not Zero mean capture is in progress, non-zero mean capture done.

See also:

e_po3030k_is_img_ready (p. 193)

9.26.3.4 char _po3030k_line_conf[330]

9.26.3.5 int _po3030k_row

9.26.3.6 int blank_row_betw [static]

9.27 camera/slow_3_timer/e_timers.c File Reference

Manage camera's interrupts (three timers).

```
#include <p30f6014a.h>
#include "./../../motor_LED/e_epuck_ports.h"
#include "e_po3030k.h"
```

Functions

- void **__attribute__** ((interrupt, auto_psv))

  The HSYNC interrupt. This interrupt is called each time the Horizontal sync signal is asserted This mean we begin the a line of the picture.

- static void **init_timer5** (void)
- static void **init_timer4** (void)
- static void **init_timer1** (void)
- void **e_po3030k_launch_capture** (char ∗buf)
- void **e_po3030k_apply_timer_config** (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)
- int **e_po3030k_is_img_ready** (void)

Variables

- static int **max_row**
- static int **max_col**
- static char ∗ **buffer**
- static int **img_ready**
- static int **current_row**
- static int **current_col**
- static int **buf_pos**
- static int **bytes_per_pixel**
- static int **bpp_current**
- static int **pixel_betw_pixel**
- static int **pbp_current**
- static int **blank_betw_lines**
- static int **bbl_current**

9.27.1 Detailed Description

Manage camera's interrupts (three timers).

9.27.2 Function Documentation

9.27.2.1 void __attribute__ ((interrupt, auto_psv))

The HSYNC interrupt. This interrupt is called each time the Horizontal sync signal is asserted This mean we begin the a line of the picture.

The Pixel Clock interrupt. This interrupt is called every time the Pixel clock signal is asserted This mean that the next byte is ready to be read.

The VSYNC interrupt. This interrupt is called every time the Vertical sync signal is asserted This mean that the picture is comming from the camera (we will have the first line soon).

9.27.2.2 void e_po3030k_apply_timer_config (int *pixel_row*, int *pixel_col*, int *bpp*, int *pbp*, int *bbl*)

Modify the interrupt configuration

Warning:

This is an internal function, use *e_po3030k_config_cam*

Parameters:

pixel_row The number of row to take

pixel_col The number of pixel to take each *pixel_row*

bpp The number of byte per pixel

pbp The number of pixel to ignore between each pixel

bbl The number of row to ignore between each line

See also:

e_po3030k_get_bytes_per_pixel (p. 145) and **e_po3030k_config_cam** (p. 144)

9.27.2.3 int e_po3030k_is_img_ready (void)

Check if the current capture is finished

Returns:

Zero if the current capture is in progress, non-zero if the capture is done.

See also:

e_po3030k_launch_capture (p. 193)

9.27.2.4 void e_po3030k_launch_capture (char * *buf*)

Launch a capture in the *buf* buffer

Parameters:

buf The buffer to write to

See also:

e_po3030k_config_cam (p. 144) and **e_po3030k_is_img_ready** (p. 193)

9.27.2.5 static void init_timer1 (void) [static]

9.27.2.6 static void init_timer4 (void) [static]

9.27.2.7 static void init_timer5 (void) [static]

9.27.3 Variable Documentation

9.27.3.1 int bbl_current [static]

The current row we ignore between each effective row

9.27.3.2 int blank_betw_lines [static]

The number of blank row between each effective row

9.27.3.3 int bpp_current [static]

The current byte we are inside a pixel

9.27.3.4 int buf_pos [static]

Counter which is incremented each time we acquire a byte

9.27.3.5 char* buffer [static]

The buffer to write to

9.27.3.6 int bytes_per_pixel [static]

Number of bytes per pixel

9.27.3.7 int current_col [static]

The current column we are

9.27.3.8 int current_row [static]

The current row we are

9.27.3.9 int img_ready [static]

The flag to tell, the image is ready or not Zero mean capture is in progress, non-zero mean capture done.

See also:

e_po3030k_is_img_ready (p. 193)

9.27.3.10 int max_col [static]

The number of bytes per row we should take

See also:

e_po3030k_get_bytes_per_pixel (p. 145), **e_po3030k_apply_timer_config** (p. 193)

9.27.3.11 int max_row [static]

The number of row we should take

See also:

e_po3030k_aApply_timer_config

9.27.3.12 int pbp_current [static]

The current pixel we are between two effective pixel

9.27.3.13 int pixel_betw_pixel [static]

Pixel to "jump" between each effective pixel

9.28 codec/e_common.inc File Reference

9.29 codec/e_sound.c File Reference

Package to play basics sounds on the e-puck's speaker.

For more info look at this: **Sound** (p. 21).

```
#include "../motor_led/e_epuck_ports.h"
#include "e_sound.h"
```

Functions

- void **e_init_sound** (void)

  Initialize all you need to play sound on speaker.

- void **e_play_sound** (int sound_nbr, int sound_length)

  Play a sound.

- void **e_close_sound** (void)

  Disable the sound After that you can't play sound anymore, if you want to, you have to call e_init_sound.

9.29.1 Detailed Description

Package to play basics sounds on the e-puck's speaker.

For more info look at this: **Sound** (p. 21).

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.29.2 Function Documentation

9.29.2.1 void e_close_sound (void)

Disable the sound After that you can't play sound anymore, if you want to, you have to call e_init_sound.

9.29.2.2 void e_init_sound (void)

Initialize all you need to play sound on speaker.

Warning:

You must to call this function before playing a sound (call it only one time).

9.29.2.3 void e_play_sound (int *sound_nbr*, int *sound_length*)

Play a sound.

Parameters:

sound_nbr the begining of the sound

sound_length the length of the sound

See also:

Sound (p. 21)

9.30 codec/e_sound.h File Reference

Package to play basics sounds on the e-puck's speaker.

Functions

- void **e_init_sound** (void)

  Initialize all you need to play sound on speaker.

- void **e_play_sound** (int sound_offset, int sound_length)

  Play a sound.

- void **e_close_sound** (void)

  Disable the sound After that you can't play sound anymore, if you want to, you have to call e_init_sound.

- void **e_init_dci_master** (void)
- void **e_init_codec_slave** (void)
- void **e_sub_dci_kickoff** (int, int)

9.30.1 Detailed Description

Package to play basics sounds on the e-puck's speaker.

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.30.2 Function Documentation

9.30.2.1 void e_close_sound (void)

Disable the sound After that you can't play sound anymore, if you want to, you have to call e_init_sound.

9.30.2.2 void e_init_codec_slave (void)

9.30.2.3 void e_init_dci_master (void)

9.30.2.4 void e_init_sound (void)

Initialize all you need to play sound on speaker.

Warning:

You must to call this function before playing a sound (call it only one time).

9.30.2.5 void e_play_sound (int *sound_nbr*, int *sound_length*)

Play a sound.

Parameters:

sound_nbr the begining of the sound

sound_length the length of the sound

See also:

Sound (p. 21)

9.30.2.6 void e_sub_dci_kickoff (int, int)

9.31 contrib/LIS_sensors_turret/e_devantech.c File Reference

```
#include "e_devantech.h"
#include "./../../I2C/e_I2C_master_module.h"
#include "./../../I2C/e_I2C_protocol.h"
```

Functions

- void **e_init_devantech** (void)
- void **e_disable_devantech** (void)
- unsigned int **e_get_dist_devantech** (char device_add)
- unsigned int **e_get_delay_devantech** (char device_add)
- char **e_get_sr_devantech** (char device_add)
- unsigned int **e_get_light_devantech** (char device_add)
- void **e_set_gain_devantech** (char device_add, char gain)
- void **e_set_range_devantech** (char device_add, char range)
- void **e_i2cd_write** (char device_add, char reg, char value)
- char **e_i2cd_readb** (char device_add, char reg)

9.31.1 Function Documentation

9.31.1.1 void e_disable_devantech (void)

9.31.1.2 unsigned int e_get_delay_devantech (char *device_add*)

9.31.1.3 unsigned int e_get_dist_devantech (char *device_add*)

9.31.1.4 unsigned int e_get_light_devantech (char *device_add*)

9.31.1.5 char e_get_sr_devantech (char *device_add*)

9.31.1.6 char e_i2cd_readb (char *device_add*, char *reg*)

9.31.1.7 void e_i2cd_write (char *device_add*, char *reg*, char *value*)

9.31.1.8 void e_init_devantech (void)

9.31.1.9 void e_set_gain_devantech (char *device_add*, char *gain*)

9.31.1.10 void e_set_range_devantech (char *device_add*, char *range*)

9.32 contrib/LIS_sensors_turret/e_devantech.h File Reference

Devantech sensor of e-puck.

```
#include "./../../I2C/e_I2C_master_module.h"
#include "./../../motor_led/e_epuck_ports.h"
```

Functions

- void **e_init_devantech** (void)
- void **e_disable_devantech** (void)
- unsigned int **e_get_dist_devantech** (char device_add)
- unsigned int **e_get_delay_devantech** (char device_add)
- char **e_get_sr_devantech** (char device_add)
- unsigned int **e_get_light_devantech** (char device_add)
- void **e_set_gain_devantech** (char device_add, char gain)
- void **e_set_range_devantech** (char device_add, char range)
- void **e_i2cd_write** (char device_add, char reg, char value)
- char **e_i2cd_readb** (char device_add, char reg)
- unsigned int **e_i2cd_readw** (char device_add, char reg)

9.32.1 Detailed Description

Devantech sensor of e-puck.

Author:

Jonathan Besuchet

9.32.2 Function Documentation

9.32.2.1 void e_disable_devantech (void)

9.32.2.2 unsigned int e_get_delay_devantech (char *device_add*)

9.32.2.3 unsigned int e_get_dist_devantech (char *device_add*)

9.32.2.4 unsigned int e_get_light_devantech (char *device_add*)

9.32.2.5 char e_get_sr_devantech (char *device_add*)

9.32.2.6 char e_i2cd_readb (char *device_add*, char *reg*)

9.32.2.7 unsigned int e_i2cd_readw (char *device_add*, char *reg*)

9.32.2.8 void e_i2cd_write (char *device_add*, char *reg*, char *value*)

9.32.2.9 void e_init_devantech (void)

9.32.2.10 void e_set_gain_devantech (char *device_add*, char *gain*)

9.32.2.11 void e_set_range_devantech (char *device_add*, char *range*)

9.33 contrib/LIS_sensors_turret/e_sensext.c File Reference

```
#include "e_sensext.h"
#include "e_sharp.h"
#include <a_d/advance_ad_scan/e_acc.h>
#include <motor_led/advance_one_timer/e_agenda.h>
#include <I2C/e_I2C_master_module.h>
#include <I2C/e_I2C_protocol.h>
```

Functions

- void **e_stop_sensext_wait** (void)
- void **e_start_sensext_wait** (void)
- int **e_get_sensext_wait** (void)
- void **e_init_sensext** (void)
- int **e_sensext_process** (int *sensext_param, unsigned int *sensext_value)

Variables

- static int **sensext_wait** = 0

9.33.1 Function Documentation

9.33.1.1 int e_get_sensext_wait (void)

9.33.1.2 void e_init_sensext (void)

9.33.1.3 int e_sensext_process (int * *sensext_param*, unsigned int * *sensext_value*)

!!WALTER!!!!!

!!WALTER!!!!!

9.33.1.4 void e_start_sensext_wait (void)

9.33.1.5 void e_stop_sensext_wait (void)

9.33.2 Variable Documentation

9.33.2.1 int sensext_wait = 0 [static]

9.34 contrib/LIS_sensors_turret/e_sensext.h File Reference

Defines

- #define **I2C_ADDR_SENSEXT** 0b10100010
- #define **I2C_ADDR_SRF08** 0xE0
- #define **I2C_ADDR_SRF10** 0xE0
- #define **I2C_ADDR_CMPS03** 0xC0
- #define **I2C_ADDR_SRF235** 0xE0

Functions

- void **e_init_sensext** (void)
- int **e_sensext_process** (int ∗sensext_param, unsigned int ∗sensext_value)
- void **e_stop_sensext_wait** (void)
- void **e_start_sensext_wait** (void)
- int **e_get_sensext_wait** (void)

9.34.1 Define Documentation

9.34.1.1 #define I2C_ADDR_CMPS03 0xC0

9.34.1.2 #define I2C_ADDR_SENSEXT 0b10100010

9.34.1.3 #define I2C_ADDR_SRF08 0xE0

9.34.1.4 #define I2C_ADDR_SRF10 0xE0

9.34.1.5 #define I2C_ADDR_SRF235 0xE0

9.34.2 Function Documentation

9.34.2.1 int e_get_sensext_wait (void)

9.34.2.2 void e_init_sensext (void)

9.34.2.3 int e_sensext_process (int ∗ *sensext_param*, unsigned int ∗ *sensext_value*)

!!WALTER!!!!!

!!WALTER!!!!!

9.34.2.4 void e_start_sensext_wait (void)

9.34.2.5 void e_stop_sensext_wait (void)

9.35 contrib/LIS_sensors_turret/e_sharp.c File Reference

```
#include "a_d/e_ad_conv.h"
#include "motor_led/e_epuck_ports.h"
#include "e_sharp.h"
#include <a_d/advance_ad_scan/e_acc.h>
```

Functions

- void **e_init_sharp** (void)
- int **e_get_dist_sharp** ()
- void **e_set_sharp_led** (unsigned int sharp_led_number, unsigned int value)
- void **e_sharp_led_clear** (void)
- void **e_sharp_on** (void)
- void **e_sharp_off** (void)

9.35.1 Function Documentation

9.35.1.1 int e_get_dist_sharp (void)

9.35.1.2 void e_init_sharp (void)

9.35.1.3 void e_set_sharp_led (unsigned int *sharp_led_number*, unsigned int *value*)

9.35.1.4 void e_sharp_led_clear (void)

9.35.1.5 void e_sharp_off (void)

9.35.1.6 void e_sharp_on (void)

9.36 contrib/LIS_sensors_turret/e_sharp.h File Reference

Defines

- #define **SHARP** 5
- #define **SHARP_LED1** _LATG6
- #define **SHARP_LED2** _LATG7
- #define **SHARP_LED3** _LATG8
- #define **SHARP_LED4** _LATG9
- #define **SHARP_LED5** _LATB6
- #define **SHARP_VIN** _LATB7
- #define **SHARP_LED1_DIR** _TRISG6
- #define **SHARP_LED2_DIR** _TRISG7
- #define **SHARP_LED3_DIR** _TRISG8
- #define **SHARP_LED4_DIR** _TRISG9
- #define **SHARP_LED5_DIR** _TRISB6
- #define **SHARP_VIN_DIR** _TRISB7

Functions

- void **e_init_sharp** (void)
- int **e_get_dist_sharp** (void)
- void **e_set_sharp_led** (unsigned int sharp_led_number, unsigned int value)
- void **e_sharp_led_clear** (void)
- void **e_sharp_on** (void)
- void **e_sharp_off** (void)

9.36.1 Define Documentation

9.36.1.1 #define SHARP 5

9.36.1.2 #define SHARP_LED1 _LATG6

9.36.1.3 #define SHARP_LED1_DIR _TRISG6

9.36.1.4 #define SHARP_LED2 _LATG7

9.36.1.5 #define SHARP_LED2_DIR _TRISG7

9.36.1.6 #define SHARP_LED3 _LATG8

9.36.1.7 #define SHARP_LED3_DIR _TRISG8

9.36.1.8 #define SHARP_LED4 _LATG9

9.36.1.9 #define SHARP_LED4_DIR _TRISG9

9.36.1.10 #define SHARP_LED5 _LATB6

9.36.1.11 #define SHARP_LED5_DIR _TRISB6

9.36.1.12 #define SHARP_VIN _LATB7

9.36.1.13 #define SHARP_VIN_DIR _TRISB7

9.36.2 Function Documentation

9.36.2.1 int e_get_dist_sharp (void)

9.36.2.2 void e_init_sharp (void)

9.36.2.3 void e_set_sharp_led (unsigned int *sharp_led_number*, unsigned int *value*)

9.36.2.4 void e_sharp_led_clear (void)

9.36.2.5 void e_sharp_off (void)

9.36.2.6 void e_sharp_on (void)

9.37 contrib/SWIS_com_module/ComModule.c File Reference

```
#include "ComModule.h"
#include <stdlib.h>
#include "./motor_led/e_epuck_ports.h"
#include "./I2C/e_I2C_protocol.h"
#include "./I2C/e_I2C_master_module.h"
```

Defines

- #define **COM_MODULE_I2C_ADDR** (0x3F<<1)
- #define **BUFFER_DATA_LENGTH** (COM_MODULE_MAXSIZE + 14)
- #define **STATUS_REG_ADDR** 0x00
- #define **CONFIG_REG_ADDR** 0x01
- #define **SEND_REG_ADDR** 0x02
- #define **SOFTATT_REG_ADDR** 0x03
- #define **OWNGROUP_REG_ADDR** 0x04
- #define **OWNADDRL_REG_ADDR** 0x05
- #define **OWNADDRH_REG_ADDR** 0x06
- #define **SEND_BUFFER_START** 0x07
- #define **SEND_BUFFER_END** (SEND_BUFFER_START + BUFFER_DATA_LENGTH)
- #define **REC_BUFFER_START** (SEND_BUFFER_END + 1)
- #define **REC_BUFFER_END** (REC_BUFFER_START + BUFFER_DATA_LENGTH)
- #define **AM_MSGTYPE** 0x0A
- #define **AM_MSGTYPE_IN_PACKET_OFFSET** 0x08
- #define **ADDRLDATA_IN_PACKET_OFFSET** 0x06
- #define **ADDRHDATA_IN_PACKET_OFFSET** 0x07
- #define **TYPEDATA_IN_PACKET_OFFSET** 0x08
- #define **GROUPDATA_IN_PACKET_OFFSET** 0x09
- #define **FIRSTDATA_IN_PACKET_OFFSET** 0x0A
- #define **PACKET_READY_FLAG** 0x01
- #define **TX_IDLE_FLAG** 0x02
- #define **PACKET_LOST_FLAG** 0x04
- #define **TX_SEND_ERROR** 0x08
- #define **HARDWAREATT_SET_FLAG** 0x01
- #define **RADIO_ENABLED_FLAG** 0x80
- #define **REQUEST_TO_SEND_FLAG** 0x01

Functions

- unsigned char **ReadRegister** (unsigned char registeraddr)
- void **WriteRegister** (unsigned char registeraddr, unsigned char value)
- void **InitComModule** (unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue)
- int **IsModulePlugged** ()
- void **SetRadioEnabledState** (unsigned char mode)
- void **SetHardwareAttenuator** (unsigned char attenuatormode)
- void **SetSoftwareAttenuator** (unsigned char attenuatorvalue)

- void **SetOwnGroup** (unsigned char owngroup)

- void **SetOwnAddress** (unsigned int ownaddress)

- unsigned char **GetHardwareAttenuator** ()

- unsigned char **GetRadioEnabledState** ()

- unsigned char **GetSoftwareAttenuator** ()

- unsigned char **GetOwnGroup** ()

- unsigned int **GetOwnAddress** ()

- unsigned char **GetStatus** ()

- int **SendPacket** (unsigned char destinationgroup, unsigned int destinationaddress, unsigned char *packet, int packetsize)

- int **IsPacketReady** (unsigned char *packet, int *packetsize)

9.37.1 Define Documentation

9.37.1.1 #define ADDRHDATA_IN_PACKET_OFFSET 0x07

9.37.1.2 #define ADDRLDATA_IN_PACKET_OFFSET 0x06

9.37.1.3 #define AM_MSGTYPE 0x0A

9.37.1.4 #define AM_MSGTYPE_IN_PACKET_OFFSET 0x08

9.37.1.5 #define BUFFER_DATA_LENGTH (COM_MODULE_MAXSIZE + 14)

9.37.1.6 #define COM_MODULE_I2C_ADDR (0x3F<<1)

9.37.1.7 #define CONFIG_REG_ADDR 0x01

9.37.1.8 #define FIRSTDATA_IN_PACKET_OFFSET 0x0A

9.37.1.9 #define GROUPDATA_IN_PACKET_OFFSET 0x09

9.37.1.10 #define HARDWAREATT_SET_FLAG 0x01

9.37.1.11 #define OWNADDRH_REG_ADDR 0x06

9.37.1.12 #define OWNADDRL_REG_ADDR 0x05

9.37.1.13 #define OWNGROUP_REG_ADDR 0x04

9.37.1.14 #define PACKET_LOST_FLAG 0x04

9.37.1.15 #define PACKET_READY_FLAG 0x01

9.37.1.16 #define RADIO_ENABLED_FLAG 0x80

9.37.1.17 #define REC_BUFFER_END (REC_BUFFER_START + BUFFER_DATA_LENGTH)

9.37.1.18 #define REC_BUFFER_START (SEND_BUFFER_END + 1)

9.37.1.19 #define REQUEST_TO_SEND_FLAG 0x01

9.37.1.20 #define SEND_BUFFER_END (SEND_BUFFER_START + BUFFER_DATA_LENGTH)

9.37.1.21 #define SEND_BUFFER_START 0x07

9.37.1.22 #define SEND_REG_ADDR 0x02

9.37.1.23 #define SOFTATT_REG_ADDR 0x03

9.37.1.24 #define STATUS_REG_ADDR 0x00

9.37.1.25 #define TX_IDLE_FLAG 0x02

9.37.1.26 #define TX_SEND_ERROR 0x08

9.37.1.27 #define TYPEDATA_IN_PACKET_OFFSET 0x08

9.37.2 Function Documentation

9.37.2.1 unsigned char GetHardwareAttenuator ()

9.37.2.2 unsigned int GetOwnAddress ()

9.38 contrib/SWIS_com_module/ComModule.h File Reference

Radio communication.

```
#include "p30f6014A.h"
```

Defines

- #define **COM_MODULE_HW_ATTENUATOR_25DB** 1
- #define **COM_MODULE_HW_ATTENUATOR_0DB** 0
- #define **COM_MODULE_DEFAULT_GROUP** 0x7d
- #define **COM_MODULE_MAXSIZE** 108

Functions

- void **InitComModule** (unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue)
- int **IsModulePlugged** ()
- void **SetRadioEnabledState** (unsigned char mode)
- void **SetHardwareAttenuator** (unsigned char AttenuatorMode)
- void **SetSoftwareAttenuator** (unsigned char AttenuatorValue)
- void **SetOwnGroup** (unsigned char GroupID)
- void **SetOwnAddress** (unsigned int ownaddress)
- unsigned char **GetRadioEnabledState** ()
- unsigned char **GetHardwareAttenuator** ()
- unsigned char **GetSoftwareAttenuator** ()
- unsigned char **GetOwnGroup** ()
- unsigned char **GetStatus** ()
- int **SendPacket** (unsigned char destinationgroup, unsigned int destinationaddress, unsigned char *packet, int packetsize)
- int **IsPacketReady** (unsigned char *packet, int *packetSize)

9.38.1 Detailed Description

Radio communication.

Author:

Jonathan Besuchet

9.38.2 Define Documentation

9.38.2.1 #define COM_MODULE_DEFAULT_GROUP 0x7d

9.38.2.2 #define COM_MODULE_HW_ATTENUATOR_0DB 0

9.38.2.3 #define COM_MODULE_HW_ATTENUATOR_25DB 1

9.38.2.4 #define COM_MODULE_MAXSIZE 108

9.38.3 Function Documentation

9.38.3.1 unsigned char GetHardwareAttenuator ()

9.38.3.2 unsigned char GetOwnGroup ()

9.38.3.3 unsigned char GetRadioEnabledState ()

9.38.3.4 unsigned char GetSoftwareAttenuator ()

9.38.3.5 unsigned char GetStatus ()

9.38.3.6 void InitComModule (unsigned char *owngroup*, unsigned int *ownaddress*, unsigned char *hardwareattenuatormode*, unsigned char *softwareattenuatorvalue*)

9.38.3.7 int IsModulePlugged ()

9.38.3.8 int IsPacketReady (unsigned char * *packet*, int * *packetSize*)

9.38.3.9 int SendPacket (unsigned char *destinationgroup*, unsigned int *destinationaddress*, unsigned char * *packet*, int *packetsize*)

9.38.3.10 void SetHardwareAttenuator (unsigned char *AttenuatorMode*)

9.38.3.11 void SetOwnAddress (unsigned int *ownaddress*)

9.38.3.12 void SetOwnGroup (unsigned char *GroupID*)

9.38.3.13 void SetRadioEnabledState (unsigned char *mode*)

9.38.3.14 void SetSoftwareAttenuator (unsigned char *AttenuatorValue*)

9.39 fft/e_fft.c File Reference

Package to mange the FFT.

```
#include <dsp.h>
#include "e_fft.h"
```

Functions

- const fractcomplex twiddleFactors[FFT_BLOCK_LENGTH/2] **__attribute__** ((space(auto_psv), aligned(FFT_BLOCK_LENGTH ∗2)))
- void **e_doFFT_asm** (fractcomplex ∗sigCmpx)

  Execute the FFT with dsPic special instructions.

9.39.1 Detailed Description

Package to mange the FFT.

Author:

Code & doc: Jonathan Besuchet

9.39.2 Function Documentation

9.39.2.1 const fractcomplex twiddleFactors [FFT_BLOCK_LENGTH/2] __attribute__ ((space(auto_psv), aligned(FFT_BLOCK_LENGTH ∗2)))

9.39.2.2 void e_doFFT_asm (fractcomplex ∗ *sigCmpx*)

Execute the FFT with dsPic special instructions.

Parameters:

sigCmpx The pointer of the begining of the array on which you want to perform the FFT.

9.40 fft/e_fft.h File Reference

Package to mange the FFT.

```
#include <dsp.h>
```

Defines

- #define **FFT_BLOCK_LENGTH** 256
- #define **LOG2_BLOCK_LENGTH** 8

Functions

- void **e_doFFT_asm** (fractcomplex *sigCmpx)

  Execute the FFT with dsPic special instructions.

9.40.1 Detailed Description

Package to mange the FFT.

In this package the FFT (fast fourier transform) is done with the special dsPic 30fxxxx instructions.

Before calling the **e_doFFT_asm()** (p. 214) function, you must to fill the sigCmpx array with your values. This array is stocked in the Y memory of the dsPic.

Author:

Code & doc: Jonathan Besuchet

9.40.2 Define Documentation

9.40.2.1 #define FFT_BLOCK_LENGTH 256

9.40.2.2 #define LOG2_BLOCK_LENGTH 8

9.40.3 Function Documentation

9.40.3.1 void e_doFFT_asm (fractcomplex * *sigCmpx*)

Execute the FFT with dsPic special instructions.

Parameters:

sigCmpx The pointer of the begining of the array on which you want to perform the FFT.

9.41 fft/e_fft_utilities.h File Reference

Some fft features.

Functions

- void **e_fast_copy** (int *in_array, int *out_array, int **size**)
  Copy an array to an other array, using REPEAT instruction.
- void **e_subtract_mean** (int *array, int **size**, int log2size)
  Substract the mean from the samples of the array (-> produce zero-mean samples).

9.41.1 Detailed Description

Some fft features.

Here you have two functions that are really usefull to work with the FFT

Author:

Code & doc: Jonathan Besuchet

9.41.2 Function Documentation

9.41.2.1 void e_fast_copy (int * *in_array*, int * *out_array*, int *size*)

Copy an array to an other array, using REPEAT instruction.

Parameters:

in_array The array from which you want to copy

out_array The destination array

size The number of element you want to copy

9.41.2.2 void e_subtract_mean (int * *array*, int *size*, int *log2size*)

Substract the mean from the samples of the array (-> produce zero-mean samples).

Parameters:

array The array that you want to produce zero-mean samples

size The size of the array

log2size The log in base 2 of the size

9.42 fft/e_input_signal.c File Reference

Allocate memory and initialize the sigCmpx array.

```
#include <dsp.h>
#include "e_fft.h"
```

Functions

- fractcomplex sigCmpx[FFT_BLOCK_LENGTH] **__attribute__** ((section(".ydata, **data**, ymemory"), aligned(FFT_BLOCK_LENGTH ∗2 ∗2)))

9.42.1 Detailed Description

Allocate memory and initialize the sigCmpx array.

sigCmpx is the main array in which the FFT will be done. So you must to fill this array with your values to perform the FFT.

This array has the following size: 64, 128, 256, 512 depending the choice you have made in fft.h

Author:

Code & doc: Jonathan Besuchet

9.42.2 Function Documentation

9.42.2.1 fractcomplex sigCmpx [FFT_BLOCK_LENGTH] __attribute__ ((section(".ydata, data, ymemory"), aligned(FFT_BLOCK_LENGTH ∗2 ∗2)))

9.43 fft/e_twiddle_factors.c File Reference

The FFT factor from Microchip.

```
#include <dsp.h>
#include "e_fft.h"
```

Functions

- const fractcomplex twiddleFactors[ ] **__attribute__** ((space(auto_psv), aligned(FFT_BLOCK_-LENGTH ∗2)))

9.43.1 Detailed Description

The FFT factor from Microchip.

Author:

Jonathan Besuchet

9.43.2 Function Documentation

9.43.2.1 const fractcomplex twiddleFactors [ ] __attribute__ ((space(auto_psv), aligned(FFT_BLOCK_LENGTH ∗2)))

9.44 I2C/e_I2C_master_module.c File Reference

Manage I2C basics.

```
#include "e_I2C_master_module.h"
```

Functions

- void **idle_i2c** (void)
  Wait until I2C Bus is Inactive.
- char **e_i2c_init** (void)
  Initialize the microcontroller for I2C uses.
- char **e_i2c_reset** (void)
  Reset the microcontroller for I2C uses.
- char **e_i2c_enable** (void)
  Enable special I2C interrupt.
- char **e_i2c_disable** (void)
  Disable special I2C interrupt.
- char **e_i2c_start** (void)
  Make the start bit.
- char **e_i2c_restart** (void)
  Make the restart bit.
- char **e_i2c_stop** (void)
  Make the stop bit.
- char **e_i2c_ack** (void)
  Make the acknowledgement bit.
- char **e_i2c_nack** (void)
  Make the non-acknowledgement bit.
- char **e_i2c_read** (char ∗buf)
  Read the I2C input register.
- char **e_i2c_write** (char byte)
  Write on the I2C output register.
- void **__attribute__** ((__interrupt__, auto_psv))

Variables

- char **e_i2c_mode**
- int **e_interrupts** [3]

9.44.1 Detailed Description

Manage I2C basics.

This module manage the I2C basics functions (low level I2C functions).

They are made to perform the basics tasks like:

- initializing the I2C on the microcontroller
- sending the Start bit (e_i2c_start)
- sending the Restart bit (e_i2c_restart)
- sending the Stop bit (e_i2c_stop)
- sending the acknowledgement bit (e_i2c_ack)
- writing or receiving a byte (e_i2c_write, e_i2c_read)
- ...

Author:

Code: Davis Daidie
Doc: Jonathan Besuchet

9.44.2 Function Documentation

9.44.2.1 void __attribute__ ((__interrupt__, auto_psv))

9.44.2.2 char e_i2c_ack (void)

Make the acknowledgement bit.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.3 char e_i2c_disable (void)

Disable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.4 char e_i2c_enable (void)

Enable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.5 char e_i2c_init (void)

Initialize the microcontroller for I2C uses.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.6 char e_i2c_nack (void)

Make the non-acknowledgement bit.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.7 char e_i2c_read (char * *buf*)

Read the I2C input register.

Parameters:

buf A pointer to store the datas received

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.8 char e_i2c_reset (void)

Reset the microcontroller for I2C uses.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.9 char e_i2c_restart (void)

Make the restart bit.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.10 char e_i2c_start (void)

Make the start bit.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.11 char e_i2c_stop (void)

Make the stop bit.

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.12 char e_i2c_write (char *byte*)

Write on the I2C output register.

Parameters:

byte What you want to send on I2C

Returns:

1 to confirme the oparation and 0 for an error

9.44.2.13 void idle_i2c (void)

Wait until I2C Bus is Inactive.

9.44.3 Variable Documentation

9.44.3.1 char e_i2c_mode

9.44.3.2 int e_interrupts[3]

9.45 I2C/e_I2C_master_module.h File Reference

Manage I2C basics.

```
#include "p30f6014A.h"
```

Defines

- #define **START** 1
- #define **WRITE** 2
- #define **ACKNOWLEDGE** 3
- #define **READ** 4
- #define **STOP** 5
- #define **RESTART** 6
- #define **ERROR** 10
- #define **OPERATION_OK** 0

Functions

- char **e_i2c_init** (void)
  Initialize the microcontroller for I2C uses.
- char **e_i2c_start** (void)
  Make the start bit.
- char **e_i2c_restart** (void)
  Make the restart bit.
- char **e_i2c_ack** (void)
  Make the acknowledgement bit.
- char **e_i2c_nack** (void)
  Make the non-acknowledgement bit.
- char **e_i2c_read** (char ∗buf)
  Read the I2C input register.
- char **e_i2c_stop** (void)
  Make the stop bit.
- char **e_i2c_write** (char byte)
  Write on the I2C output register.
- char **e_i2c_enable** (void)
  Enable special I2C interrupt.
- char **e_i2c_disable** (void)
  Disable special I2C interrupt.
- char **e_i2c_reset** (void)
  Reset the microcontroller for I2C uses.

9.45.1 Detailed Description

Manage I2C basics.

Author:

Code: Davis Daidie
Doc: Jonathan Besuchet

9.45.2 Define Documentation

9.45.2.1 #define ACKNOWLEDGE 3

9.45.2.2 #define ERROR 10

9.45.2.3 #define OPERATION_OK 0

9.45.2.4 #define READ 4

9.45.2.5 #define RESTART 6

9.45.2.6 #define START 1

9.45.2.7 #define STOP 5

9.45.2.8 #define WRITE 2

9.45.3 Function Documentation

9.45.3.1 char e_i2c_ack (void)

Make the acknowledgement bit.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.2 char e_i2c_disable (void)

Disable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.3 char e_i2c_enable (void)

Enable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.4 char e_i2c_init (void)

Initialize the microcontroller for I2C uses.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.5 char e_i2c_nack (void)

Make the non-acknowledgement bit.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.6 char e_i2c_read (char * *buf*)

Read the I2C input register.

Parameters:

buf A pointer to store the datas received

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.7 char e_i2c_reset (void)

Reset the microcontroller for I2C uses.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.8 char e_i2c_restart (void)

Make the restart bit.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.9 char e_i2c_start (void)

Make the start bit.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.10 char e_i2c_stop (void)

Make the stop bit.

Returns:

1 to confirme the oparation and 0 for an error

9.45.3.11 char e_i2c_write (char *byte*)

Write on the I2C output register.

Parameters:

byte What you want to send on I2C

Returns:

1 to confirme the oparation and 0 for an error

9.46 I2C/e_I2C_protocol.c File Reference

Manage I2C protocole.

```
#include "e_I2C_protocol.h"
```

Functions

- void **e_i2cp_init** (void)

  Initialize the microcontroller for I2C uses.

- void **e_i2cp_enable** (void)

  Enable special I2C interrupt.

- void **e_i2cp_disable** (void)

  Disable special I2C interrupt.

- char **e_i2cp_read** (char device_add, char reg)

  Read a specific register on a device.

- char **e_i2cp_write** (char device_add, char reg, char value)

  Write a specific register on a device.

9.46.1 Detailed Description

Manage I2C protocole.

This module manage the I2C protocole. The function's module are made to directly send or receive data from or to a specified slave.

Warning:

This file must be include to communicate with an PO3030K camera throught the I2C communication protocol

Author:

Code: Davis Daidie
Doc: Jonathan Besuchet

9.46.2 Function Documentation

9.46.2.1 void e_i2cp_disable (void)

Disable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.46.2.2 void e_i2cp_enable (void)

Enable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.46.2.3 void e_i2cp_init (void)

Initialize the microcontroller for I2C uses.

Returns:

1 to confirme the oparation and 0 for an error

9.46.2.4 char e_i2cp_read (char *device_add*, char *reg*)

Read a specific register on a device.

Parameters:

device_add The address of the device you want information

reg The register address you want read on the device

Returns:

The readed value

9.46.2.5 char e_i2cp_write (char *device_add*, char *reg*, char *value*)

Write a specific register on a device.

Parameters:

device_add The address of the device you want information

reg The register address you want read on the device

value The data you want to write

Returns:

1 to confirme the oparation and 0 for an error

9.47 I2C/e_I2C_protocol.h File Reference

Manage I2C protocole.

```
#include "e_I2C_master_module.h"
#include "../motor_led/e_epuck_ports.h"
```

Functions

- void **e_i2cp_init** (void)

  Initialize the microcontroller for I2C uses.

- char **e_i2cp_write** (char device_add, char reg, char value)

  Write a specific register on a device.

- char **e_i2cp_read** (char device_add, char reg)

  Read a specific register on a device.

- char **e_i2cp_read_string** (char device_add, unsigned char read_buffer[ ], char start_address, char string_length)
- char **e_i2cp_write_string** (char device_add, unsigned char write_buffer[ ], char start_address, char string_length)
- void **e_i2cp_enable** (void)

  Enable special I2C interrupt.

- void **e_i2cp_disable** (void)

  Disable special I2C interrupt.

9.47.1 Detailed Description

Manage I2C protocole.

This module manage the I2C protocole. The function's module are made to directly send or receive data from or to a specified slave.

Warning:

This file must be include to communicate with an PO3030K camera throught the I2C communication protocol

Author:

Code: Davis Daidie
Doc: Jonathan Besuchet

9.47.2 Function Documentation

9.47.2.1 void e_i2cp_disable (void)

Disable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.47.2.2 void e_i2cp_enable (void)

Enable special I2C interrupt.

Returns:

1 to confirme the oparation and 0 for an error

9.47.2.3 void e_i2cp_init (void)

Initialize the microcontroller for I2C uses.

Returns:

1 to confirme the oparation and 0 for an error

9.47.2.4 char e_i2cp_read (char *device_add*, char *reg*)

Read a specific register on a device.

Parameters:

device_add The address of the device you want information

reg The register address you want read on the device

Returns:

The readed value

9.47.2.5 char e_i2cp_read_string (char *device_add*, unsigned char *read_buffer*[ ], char *start_address*, char *string_length*)

9.47.2.6 char e_i2cp_write (char *device_add*, char *reg*, char *value*)

Write a specific register on a device.

Parameters:

device_add The address of the device you want information

reg The register address you want read on the device

value The data you want to write

Returns:

1 to confirme the oparation and 0 for an error

9.47.2.7 char e_i2cp_write_string (char *device_add*, unsigned char *write_buffer*[ ], char *start_address*, char *string_length*)

9.48 matlab/matlab files/CloseEpuck.m File Reference

Functions

- id **EpuckPort** ()
- **fclose** (EpuckPort)

Variables

- clear **EpuckPort**

9.48.1 Function Documentation

9.48.1.1 id EpuckPort () [virtual]

9.48.1.2 fclose (EpuckPort)

9.48.2 Variable Documentation

9.48.2.1 catch could not open the delete the global variable clear EpuckPort clear global EpuckPort

9.49 matlab/matlab files/EpuckFlush.m File Reference

Functions

- id **EpuckPort** ()
- **flushinput** (EpuckPort)
- **flushoutput** (EpuckPort)

Variables

- **function** [ ]
- Matlab probably send old **data** if you don t do this **flush**

9.49.1 Function Documentation

9.49.1.1 id EpuckPort () [virtual]

Type constraints

9.49.1.2 flushinput (EpuckPort)

9.49.1.3 flushoutput (EpuckPort)

9.49.2 Variable Documentation

9.49.2.1 Matlab probably send old data if you don t do this flush

9.49.2.2 function[ ]

Initial value:

```
 EpuckFlush()
%EPUCKFLUSH Flush the input and output buffer of Matlab
%   This function is useful each time you do a send / receive cylce with
%   the e-puck. In fact
```

9.50 matlab/matlab files/EpuckGetData.m File Reference

Functions

- **while** (i∼=5) c
- switch **i** case **if** (c∼= 'E') **continue**
- end case **if** (c== 'I') **i**
- **elseif** (c== 'C') **i**
- end end end **if** (**receivedFormat**== 'I') **size**
- **elseif** (**receivedFormat**== 'C') **size**

Variables

- **function** [**data**]
- **i** = 0
- **continue**
- **receivedFormat** = c
- **data** = fread(EpuckPort,**size**/2,'int16')
- end **return**

9.50.1 Function Documentation

9.50.1.1 elseif (receivedFormat = = 'C')

9.50.1.2 elseif (c = = 'C')

9.50.1.3 end end end if (receivedFormat = = 'I')

9.50.1.4 end case if (c = = 'I')

9.50.1.5 end case if (c∼ = 'E')

9.50.1.6 while (i∼ = 5)

9.50.2 Variable Documentation

9.50.2.1 continue

9.50.2.2 data = fread(EpuckPort,size/2,'int16')

9.50.2.3 function[data]

Initial value:

```
 EpuckGetData()
global EpuckPort
```

9.50.2.4 else i = 0

9.50.2.5 receivedFormat = c

9.50.2.6 end return

9.51 matlab/matlab files/EpuckSendData.m File Reference

Functions

- **if** (strcmp(dataType,'char')) int8(**data**)
- **fwrite** (EpuckPort, **size**,'uint16')
- **fwrite** (EpuckPort, **data**,'int8')
- else **int16** (**data**)
- **fwrite** (EpuckPort, 2 ∗**size**,'uint16')
- **fwrite** (EpuckPort, **data**,'int16')

Variables

- **function** [**data**]
- **size** = length(**data**)
- end **return**

9.51.1 Function Documentation

9.51.1.1 fwrite (EpuckPort, data, 'int16')

9.51.1.2 fwrite (EpuckPort, 2 ∗ *size*, 'uint16')

9.51.1.3 fwrite (EpuckPort, data, 'int8')

9.51.1.4 fwrite (EpuckPort, size, 'uint16')

9.51.1.5 if (strcmp(dataType,'char'))

9.51.1.6 else int16 (data)

9.51.2 Variable Documentation

9.51.2.1 function[data]

Initial value:

```
 EpuckSendData(data, dataType)
global EpuckPort
```

9.51.2.2 end return

9.51.2.3 size = length(data)

9.52 matlab/matlab files/OpenEpuck.m File Reference

Functions

- **port EpuckPort** ()
- try **fopen** (EpuckPort)

Variables

- **EpuckPort** = serial(**port**,'BaudRate', 115200,'inputBuffersize',4096,'OutputBufferSize',4096,'ByteOrder','littleendian')
- catch could not open the **port**
- disp **Error**
- disp Could not open serial **port return**

9.52.1 Function Documentation

9.52.1.1 port EpuckPort () [virtual]

9.52.1.2 try fopen (EpuckPort)

9.52.2 Variable Documentation

9.52.2.1 catch could not open the delete the global variable clear EpuckPort clear global EpuckPort = serial(port,'BaudRate', 115200,'inputBuffersize',4096,'OutputBufferSize',4096,'ByteOrder','littleendian')

9.52.2.2 disp Error

9.52.2.3 catch could not open the port

9.52.2.4 disp Could not open serial port return

9.53 matlab/matlab.c File Reference

To communicate with matlab.

```
#include "matlab.h"
#include "./../motor_led/e_epuck_ports.h"
#include "./../uart/e_uart_char.h"
```

Functions

- void **e_send_int_to_matlab** (int ∗**data**, int array_size)

  The function to send int values to matlab.

- void **e_send_char_to_matlab** (char ∗**data**, int array_size)

  The function to send char values to matlab.

- int **e_receive_int_from_matlab** (int ∗**data**, int array_size)

  The function to receive int values from matlab.

- int **e_receive_char_from_matlab** (char ∗**data**, int array_size)

  The function to receive char values from matlab.

9.53.1 Detailed Description

To communicate with matlab.

This module manage the communication with matlab through bluetooth.

Author:

Code: Michael Bonani, Jonathan Besuchet, Doc: Jonathan Besuchet

9.53.2 Function Documentation

9.53.2.1 int e_receive_char_from_matlab (char ∗ *data*, int *array_size*)

The function to receive char values from matlab.

Parameters:

data The array of char data you want to fill

array_size The length of the array

Returns:

The number of char stored

9.53.2.2 int e_receive_int_from_matlab (int * *data*, int *array_size*)

The function to receive int values from matlab.

Parameters:

data The array of int data you want to fill

array_size The length of the array

Returns:

The number of int stored

9.53.2.3 void e_send_char_to_matlab (char * *data*, int *array_size*)

The function to send char values to matlab.

Parameters:

data The array of char data you want to send

array_size The length of the array

9.53.2.4 void e_send_int_to_matlab (int * *data*, int *array_size*)

The function to send int values to matlab.

Parameters:

data The array of int data you want to send

array_size The length of the array

9.54 matlab/matlab.h File Reference

To communicate with matlab.

Functions

- void **e_send_int_to_matlab** (int *∗***data**, int array_size)
  The function to send int values to matlab.
- void **e_send_char_to_matlab** (char *∗***data**, int array_size)
  The function to send char values to matlab.
- int **e_receive_int_from_matlab** (int *∗***data**, int array_size)
  The function to receive int values from matlab.
- int **e_receive_char_from_matlab** (char *∗***data**, int array_size)
  The function to receive char values from matlab.

9.54.1 Detailed Description

To communicate with matlab.

This module manage the communication with matlab through bluetooth.

Author:

Code: Michael Bonani, Jonathan Besuchet, Doc: Jonathan Besuchet

9.54.2 Function Documentation

9.54.2.1 int e_receive_char_from_matlab (char ∗ *data*, int *array_size*)

The function to receive char values from matlab.

Parameters:

data The array of char data you want to fill

array_size The length of the array

Returns:

The number of char stored

9.54.2.2 int e_receive_int_from_matlab (int ∗ *data*, int *array_size*)

The function to receive int values from matlab.

Parameters:

data The array of int data you want to fill

array_size The length of the array

Returns:

The number of int stored

9.54.2.3 void e_send_char_to_matlab (char * *data*, int *array_size*)

The function to send char values to matlab.

Parameters:

data The array of char data you want to send

array_size The length of the array

9.54.2.4 void e_send_int_to_matlab (int * *data*, int *array_size*)

The function to send int values to matlab.

Parameters:

data The array of int data you want to send

array_size The length of the array

9.55 motor_led/advance_one_timer/e_agenda.c File Reference

Manage the agendas (timer2).

```
#include "e_agenda.h"
#include "../e_epuck_ports.h"
#include <stdlib.h>
```

Defines

- #define **EXIT_OK** 1

Functions

- void **e_start_agendas_processing** (void)

  Start the agendas processing.

- void **e_end_agendas_processing** (void)

  Stop all the agendas.

- int **e_activate_agenda** (void(∗func)(void), int cycle)

  Activate an agenda.

- int **e_destroy_agenda** (void(∗func)(void))

  Destroy an agenda.

- int **e_set_agenda_cycle** (void(∗func)(void), int cycle)

  Change the cycle value of an agenda.

- int **e_reset_agenda** (void(∗func)(void))

  Reset an agenda's counter.

- int **e_pause_agenda** (void(∗func)(void))

  Pause an agenda.

- int **e_restart_agenda** (void(∗func)(void))

  Restart an agenda previously paused.

- void **__attribute__** ((interrupt, auto_psv))

  Interrupt from timer2.

Variables

- static **Agenda** ∗ **agenda_list** = 0

9.55.1 Detailed Description

Manage the agendas (timer2).

This module manage the agendas with the timer2.

An agenda is a structure made to work as chained list. It containts: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each times the timer2 has an interrupt, all the agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, his callback function is launch.

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.55.2 Define Documentation

9.55.2.1 #define EXIT_OK 1

9.55.3 Function Documentation

9.55.3.1 void __attribute__ ((interrupt, auto_psv))

Interrupt from timer2.

Parse the chained list of agenda.

Increment counter only.

Check if agenda has to be treated according to the cycle value and current counter value.

Do it for number of cycle positive or null.

Check if a service has to be activated.

9.55.3.2 int e_activate_agenda (void(∗)(void) *func*, int *cycle*)

Activate an agenda.

Activate an agenda and allocate memory for him if there isn't already an agenda with the same callback function (the agenda is active but isn't processed if he has a null cycle value).

Parameters:

func function called if the cycle value is reached by the counter

cycle cycle value in millisec/10

Returns:

EXIT_OK (p. 314) if the agenda has been created, exit the programme otherwise

9.55.3.3 int e_destroy_agenda (void(∗)(void) *func*)

Destroy an agenda.

Destroy the agenda with a given callback function.

Parameters:

func function to test

Returns:

EXIT_OK (p. 314) if the agenda has been destroyed, **AG_NOT_FOUND** (p. 319) otherwise

9.55.3.4 void e_end_agendas_processing (void)

Stop all the agendas.

Stop all the agendas by disabling Timer2

Warning:

the memory allocated for the agenda isn't freed, use **e_destroy_agenda** (p. 320)(void (∗func)(void)) for that.

See also:

e_destroy_agenda (p. 320)

9.55.3.5 int e_pause_agenda (void(∗)(void) *func*)

Pause an agenda.

Pause an agenda but do not reset its information.

Parameters:

func function to pause

Returns:

EXIT_OK (p. 314) the agenda has been paused, **AG_NOT_FOUND** (p. 319) otherwise

9.55.3.6 int e_reset_agenda (void(∗)(void) *func*)

Reset an agenda's counter.

Reset an agenda's counter with a given callback function.

Parameters:

func function to reset

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been reseted, **AG_NOT_FOUND** (p. 319) otherwise

Warning:

This function RESET the agenda, if you just want a pause tell **e_pause_agenda** (p. 321)(void (∗func)(void))

See also:

e_pause_agenda (p. 321)

9.55.3.7 int e_restart_agenda (void(∗)(void) *func*)

Restart an agenda previously paused.

Restart an agenda previously paused.

Parameters:

func function to restart

Returns:

EXIT_OK (p. 314) if he agenda has been restarted, **AG_NOT_FOUND** (p. 319) otherwise

See also:

e_pause_agenda (p. 321)

9.55.3.8 int e_set_agenda_cycle (void(∗)(void) *func*, int *cycle*)

Change the cycle value of an agenda.

Change the cycle value of an agenda with a given callback function.

Parameters:

func function to test

cycle new cycle value in millisec/10

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been modified, **AG_NOT_FOUND** (p. 319) otherwise

9.55.3.9 void e_start_agendas_processing (void)

Start the agendas processing.

Start the agendas processing by starting the Timer2.

9.55.4 Variable Documentation

9.55.4.1 Agenda∗ agenda_list = 0 [static]

pointer on the end of agenda chained list

9.56 motor_led/advance_one_timer/e_agenda.h File Reference

Manage the agendas (timer2).

Data Structures

- struct **AgendaType**

  srtuct Agenda as chained list

Defines

- #define **AG_ALREADY_CREATED** 1
- #define **AG_NOT_FOUND** 2

Typedefs

- typedef struct **AgendaType Agenda**

Functions

- void **e_start_agendas_processing** (void)

  Start the agendas processing.

- void **e_end_agendas_processing** (void)

  Stop all the agendas.

- int **e_activate_agenda** (void(∗func)(void), int cycle)

  Activate an agenda.

- int **e_destroy_agenda** (void(∗func)(void))

  Destroy an agenda.

- int **e_set_agenda_cycle** (void(∗func)(void), int cycle)

  Change the cycle value of an agenda.

- int **e_reset_agenda** (void(∗func)(void))

  Reset an agenda's counter.

- int **e_pause_agenda** (void(∗func)(void))

  Pause an agenda.

- int **e_restart_agenda** (void(∗func)(void))

  Restart an agenda previously paused.

9.56.1 Detailed Description

Manage the agendas (timer2).

This module manage the agendas with the timer2.

An agenda is a structure made to work as chained list. It containts: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each times the timer2 has an interrupt, all the agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, his callback function is launch.

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.56.2 Define Documentation

9.56.2.1 #define AG_ALREADY_CREATED 1

9.56.2.2 #define AG_NOT_FOUND 2

9.56.3 Typedef Documentation

9.56.3.1 typedef struct AgendaType Agenda

9.56.4 Function Documentation

9.56.4.1 int e_activate_agenda (void(∗)(void) *func*, int *cycle*)

Activate an agenda.

Activate an agenda and allocate memory for him if there isn't already an agenda with the same callback function (the agenda is active but isn't processed if he has a null cycle value).

Parameters:

func function called if the cycle value is reached by the counter

cycle cycle value in millisec/10

Returns:

EXIT_OK (p. 314) if the agenda has been created, exit the programme otherwise

9.56.4.2 int e_destroy_agenda (void(∗)(void) *func*)

Destroy an agenda.

Destroy the agenda with a given callback function.

Parameters:

func function to test

Returns:

EXIT_OK (p. 314) if the agenda has been destroyed, **AG_NOT_FOUND** (p. 319) otherwise

Destroy the agenda with a given callback function

Parameters:

func function to test

Returns:

int return the success of the destruction (EXIT_OK for successfull, AG_NOT_FOUND for unsuccessfull).

9.56.4.3 void e_end_agendas_processing (void)

Stop all the agendas.

Stop all the agendas by disabling Timer2

Warning:

the memory allocated for the agenda isn't freed, use **e_destroy_agenda** (p. 320)(void (∗func)(void)) for that.

See also:

e_destroy_agenda (p. 320)

9.56.4.4 int e_pause_agenda (void(∗)(void) *func*)

Pause an agenda.

Pause an agenda but do not reset its information.

Parameters:

func function to pause

Returns:

EXIT_OK (p. 314) the agenda has been paused, **AG_NOT_FOUND** (p. 319) otherwise

Pause an agenda but do not reset its information.

Parameters:

func function to pause

9.56.4.5 int e_reset_agenda (void(∗)(void) *func*)

Reset an agenda's counter.

Reset an agenda's counter with a given callback function.

Parameters:

func function to reset

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been reseted, **AG_NOT_FOUND** (p. 319) otherwise

Warning:

This function RESET the agenda, if you just want a pause tell **e_pause_agenda** (p. 321)(void (∗func)(void))

See also:

e_pause_agenda (p. 321)

Reset an agenda's counter.

Reset an agenda's counter with a given callback function

Parameters:

func function to reset

9.56.4.6 int e_restart_agenda (void(∗)(void) *func*)

Restart an agenda previously paused.

Restart an agenda previously paused.

Parameters:

func function to restart

Returns:

EXIT_OK (p. 314) if he agenda has been restarted, **AG_NOT_FOUND** (p. 319) otherwise

See also:

e_pause_agenda (p. 321)

Restart an agenda previously paused

Parameters:

func function to restart

9.56.4.7 int e_set_agenda_cycle (void(∗)(void) *func*, int *cycle*)

Change the cycle value of an agenda.

Change the cycle value of an agenda with a given callback function.

Parameters:

func function to test

cycle new cycle value in millisec/10

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been modified, **AG_NOT_FOUND** (p. 319) otherwise

9.56.4.8 void e_start_agendas_processing (void)

Start the agendas processing.

Start the agendas processing by starting the Timer2.

Start the agendas processing.

Start the agendas processing by initialising the accounting structures.

Don't activate any timer which is done by e_start_timer_processing.

See also:

e_start_timer_processing (p. 322)

9.57 motor_led/advance_one_timer/e_led.c File Reference

Manage the LEDs with blinking possibility (timer2).

```
#include "../e_epuck_ports.h"
#include "e_agenda.h"
```

Defines

- #define **LED_EFFECTS**
  For the preprocessor. Comment if you want not to use the LED effects.

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)
  turn on/off the specified LED
- void **e_led_clear** (void)
  turn off the 8 LEDs
- void **e_set_body_led** (unsigned int value)
  turn on/off the body LED
- void **e_set_front_led** (unsigned int value)
  turn on/off the front LED
- void **e_blink_led** (void)
  Change the state of all LED.
- void **e_blink_led0** (void)
  Change the state of LED0.
- void **e_blink_led1** (void)
  Change the state of LED1.
- void **e_blink_led2** (void)
  Change the state of LED2.
- void **e_blink_led3** (void)
  Change the state of LED3.
- void **e_blink_led4** (void)
  Change the state of LED4.
- void **e_blink_led5** (void)
  Change the state of LED5.
- void **e_blink_led6** (void)

Change the state of LED6.

- void **e_blink_led7** (void)
  Change the state of LED7.
- void **e_start_led_blinking** (int cycle)
  Start blinking all LED.
- void **e_stop_led_blinking** (void)
  Stop blinking all LED.
- void **e_set_blinking_cycle** (int cycle)
  Change the blinking speed.
- void **snake_led** (void)
  One led is on and turn clockwise.
- void **flow_led** (void)
  The leds go on from the front to the back and go off from the front to the back, etc.
- void **k2000_led** (void)
  The K2000 effect.
- void **right_led** (void)
  The right LED are indicating the right side.
- void **left_led** (void)
  The left LED are indicating the left side.

9.57.1 Detailed Description

Manage the LEDs with blinking possibility (timer2).

Here we use the agenda solution to make the LED blinking.

A little exemple for LEDs blinking with agenda (all LEDs blink with 100ms delay)

Warning:

this program uses the **e_blink_led(void)** (p. 270) function.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_led.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       e_init_port();
       e_activate_agenda(e_blink_led, 1000); //blink with 100ms
       e_start_agendas_processing();
       while(1) {}
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier, Jonathan Besuchet
Doc: Jonathan Besuchet

9.57.2 Define Documentation

9.57.2.1 #define LED_EFFECTS

For the preprocessor. Comment if you want not to use the LED effects.

9.57.3 Function Documentation

9.57.3.1 void e_blink_led (void)

Change the state of all LED.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.2 void e_blink_led0 (void)

Change the state of LED0.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.3 void e_blink_led1 (void)

Change the state of LED1.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.4 void e_blink_led2 (void)

Change the state of LED2.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.5 void e_blink_led3 (void)

Change the state of LED3.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.6 void e_blink_led4 (void)

Change the state of LED4.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.7 void e_blink_led5 (void)

Change the state of LED5.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.8 void e_blink_led6 (void)

Change the state of LED6.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.9 void e_blink_led7 (void)

Change the state of LED7.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.10 void e_led_clear (void)

turn off the 8 LEDs

The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn't turn off "body LED" and "front LED".

9.57.3.11 void e_set_blinking_cycle (int *cycle*)

Change the blinking speed.

This function use **e_set_agenda_cycle** (p. 322)(void (*func)(void), int cycle)

Parameters:

cycle the number of cycle we wait before launching **e_blink_led(void)** (p. 270)"

See also:

e_blink_led (p. 270), **e_set_agenda_cycle** (p. 322)

9.57.3.12 void e_set_body_led (unsigned int *value*)

turn on/off the body LED

The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.57.3.13 void e_set_front_led (unsigned int *value*)

turn on/off the front LED

The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.57.3.14 void e_set_led (unsigned int *led_number*, unsigned int *value*)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7

value 0 (off), 1 (on) otherwise change the state

Warning:

if led_number is other than 0-7, all leds are set to the indicated value.

9.57.3.15 void e_start_led_blinking (int *cycle*)

Start blinking all LED.

Parameters:

cycle the number of cycle we wait before launching **e_blink_led(void)** (p. 270)

See also:

e_blink_led (p. 270), **e_activate_agenda** (p. 319)

9.57.3.16 void e_stop_led_blinking (void)

Stop blinking all LED.

This function use **e_destroy_agenda** (p. 320)(void (∗func)(void))

See also:

e_destroy_agenda (p. 320)

9.57.3.17 void flow_led (void)

The leds go on from the front to the back and go off from the front to the back, etc.

9.57.3.18 void k2000_led (void)

The K2000 effect.

9.57.3.19 void left_led (void)

The left LED are indicating the left side.

9.57.3.20 void right_led (void)

The right LED are indicating the right side.

9.57.3.21 void snake_led (void)

One led is on and turn clockwise.

9.58 motor_led/advance_one_timer/fast_agenda/e_led.c File Reference

Manage the LEDs with blinking possibility (timer1, 2, 3).

```
#include "../../e_epuck_ports.h"
#include "e_agenda_fast.h"
```

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)
  turn on/off the specified LED
- void **e_led_clear** (void)
  turn off the 8 LEDs
- void **e_set_body_led** (unsigned int value)
  turn on/off the body LED
- void **e_set_front_led** (unsigned int value)
  turn on/off the front LED
- void **e_blink_led** (void)
  Change the state of all LED.
- void **e_blink_led0** (void)
  Change the state of LED0.
- void **e_blink_led1** (void)
  Change the state of LED1.
- void **e_blink_led2** (void)
  Change the state of LED2.
- void **e_blink_led3** (void)
  Change the state of LED3.
- void **e_blink_led4** (void)
  Change the state of LED4.
- void **e_blink_led5** (void)
  Change the state of LED5.
- void **e_blink_led6** (void)
  Change the state of LED6.
- void **e_blink_led7** (void)
  Change the state of LED7.

- void **e_start_led_blinking** (int cycle)
  Start blinking all LED.
- void **e_stop_led_blinking** (void)
  Stop blinking all LED.
- void **e_set_blinking_cycle** (int cycle)
  Change the blinking speed.

9.58.1 Detailed Description

Manage the LEDs with blinking possibility (timer1, 2, 3).

Here we use the fast agenda solution to make the LED blinking.

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada
Doc: Jonathan Besuchet

9.58.2 Function Documentation

9.58.2.1 void e_blink_led (void)

Change the state of all LED.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.2 void e_blink_led0 (void)

Change the state of LED0.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.3 void e_blink_led1 (void)

Change the state of LED1.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.4 void e_blink_led2 (void)

Change the state of LED2.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.5 void e_blink_led3 (void)

Change the state of LED3.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.6 void e_blink_led4 (void)

Change the state of LED4.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.7 void e_blink_led5 (void)

Change the state of LED5.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.8 void e_blink_led6 (void)

Change the state of LED6.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.9 void e_blink_led7 (void)

Change the state of LED7.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.58.2.10 void e_led_clear (void)

turn off the 8 LEDs

The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn't turn off "body LED" and "front LED".

9.58.2.11 void e_set_blinking_cycle (int *cycle*)

Change the blinking speed.

This function use e_set_agenda_cycle(void (∗func)(void), int cycle)

Parameters:

cycle the number of cycle we wait before launching "e_blink_led()"

See also:

e_blink_led (p. 270), **e_set_agenda_cycle** (p. 322)

9.58.2.12 void e_set_body_led (unsigned int *value*)

turn on/off the body LED

The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.58.2.13 void e_set_front_led (unsigned int *value*)

turn on/off the front LED

The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.58.2.14 void e_set_led (unsigned int *led_number*, unsigned int *value*)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7

value 0 (off), 1 (on) otherwise change the state

Warning:

if led_number is other than 0-7, all leds are set to the indicated value.

9.58.2.15 void e_start_led_blinking (int *cycle*)

Start blinking all LED.

Parameters:

cycle the number of cycle we wait before launching "e_blink_led()"

See also:

e_blink_led (p. 270), **e_activate_agenda** (p. 319)

9.58.2.16 void e_stop_led_blinking (void)

Stop blinking all LED.

This function use e_destroy_agenda(void (∗func)(void))

See also:

e_destroy_agenda (p. 320)

9.59 motor_led/e_led.c File Reference

Manage the LEDs.

A little exemple for the LEDs (all the LEDs are blinking).

```
#include "e_epuck_ports.h"
#include "e_led.h"
```

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)

  turn on/off the specified LED

- void **e_set_body_led** (unsigned int value)

  turn on/off the body LED

- void **e_set_front_led** (unsigned int value)

  turn on/off the front LED

- void **e_led_clear** (void)

  turn off the 8 LEDs

9.59.1 Detailed Description

Manage the LEDs.

A little exemple for the LEDs (all the LEDs are blinking).

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_led.h>

int main(void)
{
       e_init_port();
       while(1)
       {
              long i;
              for(i=0; i<500000; i++)
                     asm("NOP");
              e_set_led(8, 2); //switch the state of all leds
       }
}
```

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.59.2 Function Documentation

9.59.2.1 void e_led_clear (void)

turn off the 8 LEDs

The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn't turn off "body LED" and "front LED".

9.59.2.2 void e_set_body_led (unsigned int *value*)

turn on/off the body LED

The e-puck has a green LED that illuminate his body. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.59.2.3 void e_set_front_led (unsigned int *value*)

turn on/off the front LED

The e-puck has a red LED in the front. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.59.2.4 void e_set_led (unsigned int *led_number*, unsigned int *value*)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7

value 0 (off), 1 (on) otherwise change the state

Warning:

if led_number is other than 0-7, all leds are set to the indicated value.

9.60 motor_led/advance_one_timer/e_led.h File Reference

Manage the LEDs with blinking possibility (timer2).

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)
  turn on/off the specified LED
- void **e_led_clear** (void)
  turn off the 8 LEDs
- void **e_blink_led** (void)
  Change the state of all LED.
- void **e_blink_led0** (void)
  Change the state of LED0.
- void **e_blink_led1** (void)
  Change the state of LED1.
- void **e_blink_led2** (void)
  Change the state of LED2.
- void **e_blink_led3** (void)
  Change the state of LED3.
- void **e_blink_led4** (void)
  Change the state of LED4.
- void **e_blink_led5** (void)
  Change the state of LED5.
- void **e_blink_led6** (void)
  Change the state of LED6.
- void **e_blink_led7** (void)
  Change the state of LED7.
- void **e_set_body_led** (unsigned int value)
  turn on/off the body LED
- void **e_set_front_led** (unsigned int value)
  turn on/off the front LED
- void **e_start_led_blinking** (int cycle)
  Start blinking all LED.
- void **e_stop_led_blinking** (void)

Stop blinking all LED.

- void **flow_led** (void)
  The leds go on from the front to the back and go off from the front to the back, etc.
- void **snake_led** (void)
  One led is on and turn clockwise.
- void **k2000_led** (void)
  The K2000 effect.
- void **right_led** (void)
  The right LED are indicating the right side.
- void **left_led** (void)
  The left LED are indicating the left side.

9.60.1 Detailed Description

Manage the LEDs with blinking possibility (timer2).

Here we use the agenda solution to make the LED blinking.

A little exemple for LEDs blinking with agenda (all LEDs blink with 100ms delay)

Warning:

this program uses the **e_blink_led(void)** (p. 270) function.

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_led.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       e_init_port();
       e_activate_agenda(e_blink_led, 1000); //blink with 100ms
       e_start_agendas_processing();
       while(1) {}
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier, Jonathan Besuchet
Doc: Jonathan Besuchet

9.60.2 Function Documentation

9.60.2.1 void e_blink_led (void)

Change the state of all LED.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.2 void e_blink_led0 (void)

Change the state of LED0.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.3 void e_blink_led1 (void)

Change the state of LED1.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.4 void e_blink_led2 (void)

Change the state of LED2.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.5 void e_blink_led3 (void)

Change the state of LED3.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.6 void e_blink_led4 (void)

Change the state of LED4.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.7 void e_blink_led5 (void)

Change the state of LED5.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.8 void e_blink_led6 (void)

Change the state of LED6.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.9 void e_blink_led7 (void)

Change the state of LED7.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.10 void e_led_clear (void)

turn off the 8 LEDs

The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn't turn off "body LED" and "front LED".

9.60.2.11 void e_set_body_led (unsigned int *value*)

turn on/off the body LED

The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a green LED that illuminate his body. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.60.2.12 void e_set_front_led (unsigned int *value*)

turn on/off the front LED

The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a red LED in the front. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.60.2.13 void e_set_led (unsigned int *led_number*, unsigned int *value*)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7

value 0 (off), 1 (on) otherwise change the state

Warning:

if led_number is other than 0-7, all leds are set to the indicated value.

9.60.2.14 void e_start_led_blinking (int *cycle*)

Start blinking all LED.

Parameters:

cycle the number of cycle we wait before launching **e_blink_led(void)** (p. 270)

See also:

e_blink_led (p. 270), **e_activate_agenda** (p. 319)

Parameters:

cycle the number of cycle we wait before launching "e_blink_led()"

See also:

e_blink_led (p. 270), **e_activate_agenda** (p. 319)

9.60.2.15 void e_stop_led_blinking (void)

Stop blinking all LED.

This function use **e_destroy_agenda** (p. 320)(void (∗func)(void))

See also:

e_destroy_agenda (p. 320)

This function use e_destroy_agenda(void (∗func)(void))

See also:

e_destroy_agenda (p. 320)

9.60.2.16 void flow_led (void)

The leds go on from the front to the back and go off from the front to the back, etc.

9.60.2.17 void k2000_led (void)

The K2000 effect.

9.60.2.18 void left_led (void)

The left LED are indicating the left side.

9.60.2.19 void right_led (void)

The right LED are indicating the right side.

9.60.2.20 void snake_led (void)

One led is on and turn clockwise.

9.61 motor_led/advance_one_timer/fast_agenda/e_led.h File Reference

Manage the LEDs with blinking possibility (timer1, 2, 3).

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)
  turn on/off the specified LED
- void **e_led_clear** (void)
  turn off the 8 LEDs
- void **e_blink_led** (void)
  Change the state of all LED.
- void **e_blink_led0** (void)
  Change the state of LED0.
- void **e_blink_led1** (void)
  Change the state of LED1.
- void **e_blink_led2** (void)
  Change the state of LED2.
- void **e_blink_led3** (void)
  Change the state of LED3.
- void **e_blink_led4** (void)
  Change the state of LED4.
- void **e_blink_led5** (void)
  Change the state of LED5.
- void **e_blink_led6** (void)
  Change the state of LED6.
- void **e_blink_led7** (void)
  Change the state of LED7.
- void **e_set_body_led** (unsigned int value)
  turn on/off the body LED
- void **e_set_front_led** (unsigned int value)
  turn on/off the front LED
- void **e_start_led_blinking** (int cycle)
  Start blinking all LED.

- void **e_stop_led_blinking** (void)

  Stop blinking all LED.

9.61.1 Detailed Description

Manage the LEDs with blinking possibility (timer1, 2, 3).

Here we use the fast agenda solution to make the LED blinking.

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada
Doc: Jonathan Besuchet

9.61.2 Function Documentation

9.61.2.1 void e_blink_led (void)

Change the state of all LED.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.2 void e_blink_led0 (void)

Change the state of LED0.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.3 void e_blink_led1 (void)

Change the state of LED1.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.4 void e_blink_led2 (void)

Change the state of LED2.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.5 void e_blink_led3 (void)

Change the state of LED3.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.6 void e_blink_led4 (void)

Change the state of LED4.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.7 void e_blink_led5 (void)

Change the state of LED5.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.8 void e_blink_led6 (void)

Change the state of LED6.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.9 void e_blink_led7 (void)

Change the state of LED7.

Callback function for an agenda.

See also:

AgendaType (p. 57)

9.61.2.10 void e_led_clear (void)

turn off the 8 LEDs

The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn't turn off "body LED" and "front LED".

9.61.2.11 void e_set_body_led (unsigned int *value*)

turn on/off the body LED

The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a green LED that illuminate his body. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.61.2.12 void e_set_front_led (unsigned int *value*)

turn on/off the front LED

The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a red LED in the front. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.61.2.13 void e_set_led (unsigned int *led_number*, unsigned int *value*)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7

value 0 (off), 1 (on) otherwise change the state

Warning:

if led_number is other than 0-7, all leds are set to the indicated value.

9.61.2.14 void e_start_led_blinking (int *cycle*)

Start blinking all LED.

Parameters:

cycle the number of cycle we wait before launching **e_blink_led(void)** (p. 270)

See also:

e_blink_led (p. 270), **e_activate_agenda** (p. 319)

Parameters:

cycle the number of cycle we wait before launching "e_blink_led()"

See also:

e_blink_led (p. 270), **e_activate_agenda** (p. 319)

9.61.2.15 void e_stop_led_blinking (void)

Stop blinking all LED.

This function use **e_destroy_agenda** (p. 320)(void (∗func)(void))

See also:

e_destroy_agenda (p. 320)

This function use e_destroy_agenda(void (∗func)(void))

See also:

e_destroy_agenda (p. 320)

9.62 motor_led/e_led.h File Reference

Manage the LEDs.

A little exemple for the LEDs (all the LEDs are blinking).

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)
  turn on/off the specified LED
- void **e_led_clear** (void)
  turn off the 8 LEDs
- void **e_set_body_led** (unsigned int value)
  turn on/off the body LED
- void **e_set_front_led** (unsigned int value)
  turn on/off the front LED

9.62.1 Detailed Description

Manage the LEDs.

A little exemple for the LEDs (all the LEDs are blinking).

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_led.h>

int main(void)
{
       e_init_port();
       while(1)
       {
               long i;
               for(i=0; i<500000; i++)
                       asm("NOP");
               e_set_led(8, 2); //switch the state of all leds
       }
}
```

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.62.2 Function Documentation

9.62.2.1 void e_led_clear (void)

turn off the 8 LEDs

The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn't turn off "body LED" and "front LED".

9.62.2.2 void e_set_body_led (unsigned int *value*)

turn on/off the body LED

The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a green LED that illuminate his body. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.62.2.3 void e_set_front_led (unsigned int *value*)

turn on/off the front LED

The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a red LED in the front. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.62.2.4 void e_set_led (unsigned int *led_number*, unsigned int *value*)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7

value 0 (off), 1 (on) otherwise change the state

Warning:

if led_number is other than 0-7, all leds are set to the indicated value.

9.63 motor_led/advance_one_timer/e_motors.c File Reference

Manage the motors (with timer2).

```
#include "./../e_epuck_ports.h"
#include "e_agenda.h"
#include <stdlib.h>
```

Defines

- #define **POWERSAVE**
- #define **TRESHV** 650
- #define **MAXV** 601

Functions

- void **run_left_motor** (void)
- void **run_right_motor** (void)
- void **e_init_motors** (void)

  Initialize the motors's agendas.

- void **e_set_speed_left** (int motor_speed)

  Manage the left motor speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right motor speed.

- void **e_set_speed** (int linear_speed, int angular_speed)

  Manage linear/angular speed.

- int **e_get_steps_left** ()

  Give the number of left motor steps.

- void **e_set_steps_left** (int set_steps)

  Set the number of left motor steps.

- int **e_get_steps_right** ()

  Give the number of right motor steps.

- void **e_set_steps_right** (int set_steps)

  Set the number of right motor steps.

Variables

- static int **left_speed** = 0
- static int **right_speed** = 0
- static int **left_motor_phase** = 0

- static int **right_motor_phase** = 0
- static int **nbr_steps_left** = 0
- static int **nbr_steps_right** = 0

9.63.1 Detailed Description

Manage the motors (with timer2).

This module manage the motors with the agenda solution (timer2).

A little exemple to use the motors with agenda (e-puck turn on himself)

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_motors.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       e_init_port();
       e_init_motors();
       e_set_speed(-500, 500);
       e_start_agendas_processing();
       while(1) {}
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.63.2 Define Documentation

9.63.2.1 #define MAXV 601

9.63.2.2 #define POWERSAVE

9.63.2.3 #define TRESHV 650

9.63.3 Function Documentation

9.63.3.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.63.3.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.63.3.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call **e_activate_agenda** (p. 319)(void (∗func)(void), int cycle) function.

See also:

e_activate_agenda (p. 319)

9.63.3.4 void e_set_speed (int *linear_speed*, int *angular_speed*)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

linear_speed the speed in the axis of e-puck

angular_speed the rotation speed (trigonometric)

9.63.3.5 void e_set_speed_left (int *motor_speed*)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.63.3.6 void e_set_speed_right (int *motor_speed*)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.63.3.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.63.3.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.63.3.9 void run_left_motor (void)

Change left motor phase according to the left_speed sign.

9.63.3.10 void run_right_motor (void)

Change right motor phase according to the right_speed sign

9.63.4 Variable Documentation

9.63.4.1 int left_motor_phase = 0 [static]

9.63.4.2 int left_speed = 0 [static]

9.63.4.3 int nbr_steps_left = 0 [static]

9.63.4.4 int nbr_steps_right = 0 [static]

9.63.4.5 int right_motor_phase = 0 [static]

9.63.4.6 int right_speed = 0 [static]

9.64 motor_led/advance_one_timer/fast_agenda/e_motors.c File Reference

Manage the motors (with timer1, 2, 3).

```
#include "../../e_epuck_ports.h"
#include "e_agenda_fast.h"
#include <stdlib.h>
```

Defines

- #define **POWERSAVE**
- #define **TRESHV** 650
- #define **MAXV** 601

Functions

- void **run_left_motor** (void)
- void **run_right_motor** (void)
- void **e_init_motors** (void)

  Initialize the motors's agendas.

- void **e_set_speed_left** (int motor_speed)

  Manage the left motor speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right motor speed.

- void **e_set_speed** (int linear_speed, int angular_speed)

  Manage linear/angular speed.

- int **e_get_steps_left** ()

  Give the number of left motor steps.

- void **e_set_steps_left** (int set_steps)

  Set the number of left motor steps.

- int **e_get_steps_right** ()

  Give the number of right motor steps.

- void **e_set_steps_right** (int set_steps)

  Set the number of right motor steps.

Variables

- static int **left_speed** = 0
- static int **right_speed** = 0
- static int **left_motor_phase** = 0
- static int **right_motor_phase** = 0
- static int **nbr_steps_left** = 0
- static int **nbr_steps_right** = 0

9.64.1 Detailed Description

Manage the motors (with timer1, 2, 3).

This module manage the motors with the fast agenda solution (timer1, 2, 3).

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.64.2 Define Documentation

9.64.2.1 #define MAXV 601

9.64.2.2 #define POWERSAVE

9.64.2.3 #define TRESHV 650

9.64.3 Function Documentation

9.64.3.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.64.3.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.64.3.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (∗func)(void), int cycle)" function.

See also:

e_activate_agenda (p. 319)

9.64.3.4 void e_set_speed (int *linear_speed*, int *angular_speed*)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

linear_speed the speed in the axis of e-puck

angular_speed the rotation speed (trigonometric)

9.64.3.5 void e_set_speed_left (int *motor_speed*)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.64.3.6 void e_set_speed_right (int *motor_speed*)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.64.3.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.64.3.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.64.3.9 void run_left_motor (void)

Change left motor phase according to the left_speed signe.

9.64.3.10 void run_right_motor (void)

Change right motor phase according to the right_speed signe

9.64.4 Variable Documentation

9.64.4.1 int left_motor_phase = 0 [static]

9.64.4.2 int left_speed = 0 [static]

9.64.4.3 int nbr_steps_left = 0 [static]

9.64.4.4 int nbr_steps_right = 0 [static]

9.64.4.5 int right_motor_phase = 0 [static]

9.64.4.6 int right_speed = 0 [static]

9.65 motor_led/e_motors.c File Reference

Manage the motors (with timer 4 and 5).

```
#include <stdlib.h>
#include "e_epuck_ports.h"
#include "e_init_port.h"
#include "e_motors.h"
```

Functions

- void **__attribute__** ((interrupt, auto_psv, shadow))
- void **e_init_motors** (void)

  Initialize the motors's ports.

- void **e_set_speed_left** (int motor_speed)

  Manage the left speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right speed.

- int **e_get_steps_left** (void)

  Give the number of left motor steps.

- void **e_set_steps_left** (int set_steps)

  Set the number of left motor steps.

- int **e_get_steps_right** (void)

  Give the number of right motor steps.

- void **e_set_steps_right** (int set_steps)

  Set the number of right motor steps.

Variables

- static int **left_speed** = 0
- static int **right_speed** = 0
- static int **nbr_pas_left** = 0
- static int **nbr_pas_right** = 0

9.65.1 Detailed Description

Manage the motors (with timer 4 and 5).

This module manage the motors with two timers: timer4 (motor left) and timer5 (motor right).

Warning:

You can't use this module to control the motors if you are using the camera, because the camera's module also use timer4 and timer5.

A little exemple for the motors (e-puck turn on himself)

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_motors.h>

int main(void)
{
       e_init_motors();
       e_set_speed_left(500); //go forward on half speed
       e_set_speed_right(-500); //go backward on half speed
       while(1) {}
}
```

Author:

Code: Michael Bonani, Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.65.2 Function Documentation

9.65.2.1 void __attribute__ ((interrupt, auto_psv, shadow))

9.65.2.2 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.65.2.3 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.65.2.4 void e_init_motors (void)

Initialize the motors's ports.

Initialize the motors's agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

9.65.2.5 void e_set_speed_left (int *motor_speed*)

Manage the left speed.

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer5.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.65.2.6 void e_set_speed_right (int *motor_speed*)

Manage the right speed.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer4.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.65.2.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.65.2.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.65.3 Variable Documentation

9.65.3.1 int left_speed = 0 [static]

9.65.3.2 int nbr_pas_left = 0 [static]

9.65.3.3 int nbr_pas_right = 0 [static]

9.65.3.4 int right_speed = 0 [static]

9.66 motor_led/advance_one_timer/e_motors.h File Reference

Manage the motors (with timer2).

Functions

- void **e_init_motors** (void)
  Initialize the motors's agendas.
- void **e_set_speed_left** (int motor_speed)
  Manage the left motor speed.
- void **e_set_speed_right** (int motor_speed)
  Manage the right motor speed.
- void **e_set_speed** (int linear_speed, int angular_speed)
  Manage linear/angular speed.
- void **e_set_steps_left** (int steps_left)
  Set the number of left motor steps.
- void **e_set_steps_right** (int steps_right)
  Set the number of right motor steps.
- int **e_get_steps_left** ()
  Give the number of left motor steps.
- int **e_get_steps_right** ()
  Give the number of right motor steps.

9.66.1 Detailed Description

Manage the motors (with timer2).

This module manage the motors with the agenda solution (timer2).

A little exemple to use the motors with agenda (e-puck turn on himself)

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_motors.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       e_init_port();
       e_init_motors();
       e_set_speed(-500, 500);
       e_start_agendas_processing();
       while(1) {}
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.66.2 Function Documentation

9.66.2.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.66.2.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.66.2.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call **e_activate_agenda** (p. 319)(void (∗func)(void), int cycle) function.

See also:

e_activate_agenda (p. 319)

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (∗func)(void), int cycle)" function.

See also:

e_activate_agenda (p. 319)

Initialize the motors's agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

Initialize the motors's agendas.

This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

9.66.2.4 void e_set_speed (int *linear_speed*, int *angular_speed*)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

linear_speed the speed in the axis of e-puck

angular_speed the rotation speed (trigonometric)

9.66.2.5 void e_set_speed_left (int *motor_speed*)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer5.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.66.2.6 void e_set_speed_right (int *motor_speed*)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer4.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.66.2.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.66.2.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.67 motor_led/advance_one_timer/fast_agenda/e_motors.h File Reference

Manage the motors (with timer1, 2, 3).

Functions

- void **e_init_motors** (void)

  Initialize the motors's agendas.

- void **e_set_speed_left** (int motor_speed)

  Manage the left motor speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right motor speed.

- void **e_set_speed** (int linear_speed, int angular_speed)

  Manage linear/angular speed.

- void **e_set_steps_left** (int steps_left)

  Set the number of left motor steps.

- void **e_set_steps_right** (int steps_right)

  Set the number of right motor steps.

- int **e_get_steps_left** ()

  Give the number of left motor steps.

- int **e_get_steps_right** ()

  Give the number of right motor steps.

9.67.1 Detailed Description

Manage the motors (with timer1, 2, 3).

This module manage the motors with the fast agenda solution (timer1, 2, 3).

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.67.2 Function Documentation

9.67.2.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.67.2.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.67.2.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call **e_activate_agenda** (p. 319)(void (∗func)(void), int cycle) function.

See also:

e_activate_agenda (p. 319)

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (∗func)(void), int cycle)" function.

See also:

e_activate_agenda (p. 319)

Initialize the motors's agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

Initialize the motors's agendas.

This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

9.67.2.4 void e_set_speed (int *linear_speed*, int *angular_speed*)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

linear_speed the speed in the axis of e-puck

angular_speed the rotation speed (trigonometric)

9.67.2.5 void e_set_speed_left (int *motor_speed*)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer5.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.67.2.6 void e_set_speed_right (int *motor_speed*)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer4.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.67.2.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.67.2.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.68 motor_led/e_motors.h File Reference

Manage the motors (with timer 4 and 5).

Functions

- void **e_init_motors** (void)

  Initialize the motors's agendas.

- void **e_set_speed_left** (int motor_speed)

  Manage the left motor speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right motor speed.

- int **e_get_steps_left** (void)

  Give the number of left motor steps.

- int **e_get_steps_right** (void)

  Give the number of right motor steps.

- void **e_set_steps_left** (int set_steps)

  Set the number of left motor steps.

- void **e_set_steps_right** (int set_steps)

  Set the number of right motor steps.

9.68.1 Detailed Description

Manage the motors (with timer 4 and 5).

This module manage the motors with two timers: timer4 (motor left) and timer5 (motor right).

Warning:

You can't use this module to control the motors if you are using the camera, because the camera's module also use timer4 and timer5.

A little exemple for the motors (e-puck turn on himself)

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_motors.h>

int main(void)
{
       e_init_motors();
       e_set_speed_left(500); //go forward on half speed
       e_set_speed_right(-500); //go backward on half speed
       while(1) {}
}
```

Author:

Code: Michael Bonani, Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.68.2 Function Documentation

9.68.2.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.68.2.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.68.2.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call **e_activate_agenda** (p. 319)(void (*func)(void), int cycle) function.

See also:

e_activate_agenda (p. 319)

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (*func)(void), int cycle)" function.

See also:

e_activate_agenda (p. 319)

Initialize the motors's agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

Initialize the motors's agendas.

This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

9.68.2.4 void e_set_speed_left (int *motor_speed*)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer5.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.68.2.5 void e_set_speed_right (int *motor_speed*)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer4.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.68.2.6 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.68.2.7 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.69 motor_led/advance_one_timer/e_motors_kml.c File Reference

Manage the motors (with timer2).

```
#include "./../e_epuck_ports.h"
#include "e_agenda.h"
#include <stdlib.h>
```

Defines

- #define **POWERSAVE**
- #define **TRESHV** 650
- #define **MAXV** 601

Functions

- void **run_left_motor** (void)
- void **run_right_motor** (void)
- void **e_init_motors** (void)

  Initialize the motors's agendas.

- void **e_set_speed_left** (int motor_speed)

  Manage the left motor speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right motor speed.

- void **e_set_speed** (int linear_speed, int angular_speed)

  Manage linear/angular speed.

- int **e_get_steps_left** ()

  Give the number of left motor steps.

- void **e_set_steps_left** (int set_steps)

  Set the number of left motor steps.

- int **e_get_steps_right** ()

  Give the number of right motor steps.

- void **e_set_steps_right** (int set_steps)

  Set the number of right motor steps.

Variables

- static int **goal_active_left** = 0
- static int **goal_active_right** = 0
- static int **goal_steps_left** = 0

- static int **goal_steps_right** = 0
- static int **left_speed** = 0
- static int **right_speed** = 0
- static int **left_motor_phase** = 0
- static int **right_motor_phase** = 0
- static int **nbr_steps_left** = 0
- static int **nbr_steps_right** = 0

9.69.1 Detailed Description

Manage the motors (with timer2).

This module manage the motors with the agenda solution (timer2).

A little exemple to use the motors with agenda (e-puck turn on himself)

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_motors.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       e_init_port();
       e_init_motors();
       e_set_speed(-500, 500);
       e_start_agendas_processing();
       while(1) {}
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.69.2 Define Documentation

9.69.2.1 #define MAXV 601

9.69.2.2 #define POWERSAVE

9.69.2.3 #define TRESHV 650

9.69.3 Function Documentation

9.69.3.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.69.3.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.69.3.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call **e_activate_agenda** (p. 319)(void (∗func)(void), int cycle) function.

See also:

e_activate_agenda (p. 319)

9.69.3.4 void e_set_speed (int *linear_speed*, int *angular_speed*)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

linear_speed the speed in the axis of e-puck

angular_speed the rotation speed (trigonometric)

9.69.3.5 void e_set_speed_left (int *motor_speed*)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.69.3.6 void e_set_speed_right (int *motor_speed*)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function **e_set_agenda_cycle** (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.69.3.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.69.3.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.69.3.9 void run_left_motor (void)

Change left motor phase according to the left_speed sign.

9.69.3.10 void run_right_motor (void)

Change right motor phase according to the right_speed sign

9.69.4 Variable Documentation

9.69.4.1 int goal_active_left = 0 [static]

9.69.4.2 int goal_active_right = 0 [static]

9.69.4.3 int goal_steps_left = 0 [static]

9.69.4.4 int goal_steps_right = 0 [static]

9.69.4.5 int left_motor_phase = 0 [static]

9.69.4.6 int left_speed = 0 [static]

9.69.4.7 int nbr_steps_left = 0 [static]

9.69.4.8 int nbr_steps_right = 0 [static]

9.69.4.9 int right_motor_phase = 0 [static]

9.69.4.10 int right_speed = 0 [static]

9.70 motor_led/advance_one_timer/e_remote_control.c File Reference

Manage the IR receiver module (timer2).

```
#include "e_remote_control.h"
#include "../e_epuck_ports.h"
#include "e_agenda.h"
#include "e_led.h"
```

Functions

- void **e_init_remote_control** (void)

  Initialise the IR receiver ports.

- void **__attribute__** ((__interrupt__, auto_psv))
- void **e_read_remote_control** (void)

  Read the signal and stock the information.

- unsigned char **e_get_check** (void)

  Read the check bit.

- unsigned char **e_get_address** (void)

  Read the adress of the commande.

- unsigned char **e_get_data** (void)

  Read the data of the command.

Variables

- static unsigned char **address_temp** = 0
- static unsigned char **data_temp** = 0
- static unsigned char **check_temp** = 0
- static unsigned char **address** = 0
- static unsigned char **data** = 0
- static unsigned char **check** = 2

9.70.1 Detailed Description

Manage the IR receiver module (timer2).

This module manage the IR receiver with the agenda solution (timer2).

Alittle exemple to manage the IR remote (the body LED change his state when you press a button of the IR controller).

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_remote_control.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       int ir_check;
       int previous_check = 0;
       e_init_port();
       e_init_remote_control();
       e_start_agendas_processing();
       while(1)
       {
               ir_check = e_get_check();
               if(ir_check != previous_check)
                       BODY_LED = BODY_LED^1;
               previous_check = ir_check;
       }
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.70.2 Function Documentation

9.70.2.1 void __attribute__ ((__interrupt__, auto_psv))

9.70.2.2 unsigned char e_get_address (void)

Read the adress of the commande.

Returns:

adress adress part of the signal

9.70.2.3 unsigned char e_get_check (void)

Read the check bit.

Returns:

check check bit of the signal

9.70.2.4 unsigned char e_get_data (void)

Read the data of the command.

Returns:

data data part of the signal

9.70.2.5 void e_init_remote_control (void)

Initialise the IR receiver ports.

9.70.2.6 void e_read_remote_control (void)

Read the signal and stock the information.

9.70.3 Variable Documentation

9.70.3.1 unsigned char address = 0 [static]

9.70.3.2 unsigned char address_temp = 0 [static]

9.70.3.3 unsigned char check = 2 [static]

9.70.3.4 unsigned char check_temp = 0 [static]

9.70.3.5 unsigned char data = 0 [static]

9.70.3.6 unsigned char data_temp = 0 [static]

9.71 motor_led/advance_one_timer/e_remote_control.h File Reference

Manage the LEDs with blinking possibility (timer2).

Defines

- #define **BOTTOMR** 10
- #define **BOTTOMl** 11
- #define **STANDBY** 12
- #define **MUTE** 13
- #define **VOL_UP** 16
- #define **VOL_DOWN** 17
- #define **CHAN_UP** 32
- #define **CHAN_DOWN** 33
- #define **I_II** 35
- #define **OUT_AUX_1** 56

Functions

- void **e_init_remote_control** (void)

  Initialise the IR receiver ports.

- void **e_read_remote_control** (void)

  Read the signal and stock the information.

- unsigned char **e_get_check** (void)

  Read the check bit.

- unsigned char **e_get_address** (void)

  Read the adress of the commande.

- unsigned char **e_get_data** (void)

  Read the data of the command.

9.71.1 Detailed Description

Manage the LEDs with blinking possibility (timer2).

Manage the IR receiver module (timer2).

Here we use the agenda solution to make the LED blinking.

A little exemple for LEDs blinking with agenda (all LEDs blink with 100ms delay)

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_led.h>
#include <motor_led/advance_one_timer/e_agenda.h>
```

```
int main(void)
{
       e_init_port();
       e_activate_agenda(e_blink_led, 1000); //blink with 100ms
       e_start_agendas_processing();
       while(1) {}
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Valentin Longchamp
Doc: Jonathan Besuchet

This module manage the IR receiver with the agenda solution (timer2).

A little exemple to manage the IR remote (the body LED change his state when you press a button of the IR controller).

```
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_remote_control.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
       int ir_check;
       int previous_check = 0;
       e_init_port();
       e_init_remote_control();
       e_start_agendas_processing();
       while(1)
       {
               ir_check = e_get_check();
               if(ir_check != previous_check)
                       BODY_LED = BODY_LED^1;
               previous_check = ir_check;
       }
}
```

See also:

e_agenda.h (p. 245)

Author:

Jonathan Besuchet

9.71.2 Define Documentation

9.71.2.1 #define BOTTOMl 11

9.71.2.2 #define BOTTOMR 10

9.71.2.3 #define CHAN_DOWN 33

9.71.2.4 #define CHAN_UP 32

9.71.2.5 #define I_II 35

9.71.2.6 #define MUTE 13

9.71.2.7 #define OUT_AUX_1 56

9.71.2.8 #define STANDBY 12

9.71.2.9 #define VOL_DOWN 17

9.71.2.10 #define VOL_UP 16

9.71.3 Function Documentation

9.71.3.1 unsigned char e_get_address (void)

Read the adress of the commande.

Returns:

adress adress part of the signal

9.71.3.2 unsigned char e_get_check (void)

Read the check bit.

Returns:

check check bit of the signal

9.71.3.3 unsigned char e_get_data (void)

Read the data of the command.

Returns:

data data part of the signal

9.71.3.4 void e_init_remote_control (void)

Initialise the IR receiver ports.

9.71.3.5 void e_read_remote_control (void)

Read the signal and stock the information.

9.72 motor_led/advance_one_timer/fast_agenda/e_agenda_fast.c File Reference

Manage the fast agendas (timer1, 2, 3).

```
#include "e_agenda_fast.h"
#include "../../e_epuck_ports.h"
#include <stdlib.h>
```

Defines

- #define **EXIT_OK** 1

Functions

- unsigned **compute_gcd** (unsigned u, unsigned v)
- unsigned **my_ceil** (float a)
- void **e_set_timer_speed** (int timer, unsigned speed)
- int **search_best_fit** (unsigned cycle)
- void **recompute_speeds** ()
- **Agenda** ∗ **find_function** (void(∗func)(void))
- unsigned **assign_agenda** (**Agenda** ∗agenda)
- void **migrate** (int timer)
- void **e_start_agendas_processing** (void)

  Initialise the accounting structure.

- void **e_configure_timer** (int timer)

  Configure the timer(s) used.

- void **e_start_timer_processing** (int timer)

  Start the timer(s) used.

- void **e_end_agendas_processing** (int timer)

  Stop an agenda running.

- void **e_activate_agenda** (void(∗func)(void), unsigned cycle)

  Activate a fast agenda.

- void **e_activate_motors** (void(∗func1)(void), void(∗func2)(void))
- int **e_set_motor_speed** (void(∗func)(void), unsigned cycle)
- int **e_destroy_agenda** (void(∗func)(void))

  Destroy an agenda.

- int **e_set_agenda_cycle** (void(∗func)(void), unsigned cycle)
- int **e_reset_agenda** (void(∗func)(void))

  Reset an agenda.

- int **e_pause_agenda** (void(∗func)(void))

Pause an agenda.

- int **e_restart_agenda** (void(*func)(void))

  Restart an agenda previously paused.

- void **__attribute__** ((__interrupt__, auto_psv))

  Interrupt from timer1.

Variables

- static struct **AgendaList agenda_list**

9.72.1 Detailed Description

Manage the fast agendas (timer1, 2, 3).

This module manage the fast agendas with the timer1, 2, 3.

An agenda is a structure made to work as chained list. It containts: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each times the timer1, 2, 3 has an interrupt, the corresponding agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, his callback function is launch.

Author:

Code: Julien Hubert
Doc: Jonathan Besuchet

9.72.2 Define Documentation

9.72.2.1 #define EXIT_OK 1

9.72.3 Function Documentation

9.72.3.1 void __attribute__ ((__interrupt__, auto_psv))

Interrupt from timer1.

Interrupt from timer3.

Interrupt from timer2.

Parse the chained list of agenda.

Increment counter only.

Check if agenda has to be treated according to the cycle value and current counter value.

Do it for number of cycle positive or null.

Check if a service has to be activated.

9.72.3.2 unsigned assign_agenda (Agenda * *agenda*)

9.72.3.3 unsigned compute_gcd (unsigned *u*, unsigned *v*)

9.72.3.4 void e_activate_agenda (void(*)(void) *func*, unsigned *cycle*)

Activate a fast agenda.

Activate an agenda and allocate memory for him if there isn't already an agenda with the same callback function (the agenda is active but isn't processed if he have a null cycle value).

The appropriate timer is automatically selected.

Parameters:

func function called if the cycle value is reached by the counter

cycle cycle value in millisec/10

9.72.3.5 void e_activate_motors (void(*)(void) *func1*, void(*)(void) *func2*)

9.72.3.6 void e_configure_timer (int *timer*)

Configure the timer(s) used.

Configure one or all the timers to be used by the agenda

Parameters:

timer the timer's number to configure (between [0-3] with 0 configuring all of them)

9.72.3.7 int e_destroy_agenda (void(*)(void) *func*)

Destroy an agenda.

Destroy the agenda with a given callback function

Parameters:

func function to test

Returns:

int return the success of the destruction (EXIT_OK for successfull, AG_NOT_FOUND for unsuccessfull).

9.72.3.8 void e_end_agendas_processing (int *timer*)

Stop an agenda running.

Stop the agendas running on one particular timer (the memory allocated for the agenda isn't freed, use e_destroy_agenda for that).

Parameters:

timer the timer's number [0-3]. 0 stops all the timers.

See also:

e_destroy_agenda (p. 320)

9.72.3.9 int e_pause_agenda (void(∗)(void) *func*)

Pause an agenda.

Pause an agenda but do not reset its information.

Parameters:

func function to pause

9.72.3.10 int e_reset_agenda (void(∗)(void) *func*)

Reset an agenda.

Reset an agenda's counter.

Reset an agenda's counter with a given callback function

Parameters:

func function to reset

9.72.3.11 int e_restart_agenda (void(∗)(void) *func*)

Restart an agenda previously paused.

Restart an agenda previously paused

Parameters:

func function to restart

9.72.3.12 int e_set_agenda_cycle (void(∗)(void) *func*, unsigned *cycle*)

9.72.3.13 int e_set_motor_speed (void(∗)(void) *func*, unsigned *cycle*)

9.72.3.14 void e_set_timer_speed (int *timer*, unsigned *speed*)

9.72.3.15 void e_start_agendas_processing (void)

Initialise the accounting structure.

Start the agendas processing.

Start the agendas processing by initialising the accounting structures.

Don't activate any timer which is done by e_start_timer_processing.

See also:

e_start_timer_processing (p. 322)

9.72.3.16 void e_start_timer_processing (int *timer*)

Start the timer(s) used.

Activate one or all the timers to be used by the agenda.

Parameters:

timer the timer's number to activate (between [0-3] with 0 activating all of them)

9.72.3.17 Agenda* find_function (void(*)(void) *func*)

9.72.3.18 void migrate (int *timer*)

9.72.3.19 unsigned my_ceil (float *a*)

9.72.3.20 void recompute_speeds ()

9.72.3.21 int search_best_fit (unsigned *cycle*)

9.72.4 Variable Documentation

9.72.4.1 struct AgendaList agenda_list [static]

9.73 motor_led/advance_one_timer/fast_agenda/e_agenda_fast.h File Reference

Manage the fast agendas (timer1, 2, 3).

Data Structures

- struct **AgendaType**
  srtuct Agenda as chained list
- struct **AgendaList**
  Manage the differents agendas lists.

Defines

- #define **AG_ALREADY_CREATED** 1
- #define **AG_NOT_FOUND** 2

Typedefs

- typedef struct **AgendaType Agenda**

Functions

- void **e_start_agendas_processing** (void)
  Start the agendas processing.
- void **e_start_timer_processing** (int timer)
  Start the timer(s) used.
- void **e_end_agendas_processing** (int timer)
  Stop an agenda running.
- void **e_configure_timer** (int timer)
  Configure the timer(s) used.
- void **e_activate_agenda** (void(∗func)(void), unsigned cycle)
  Activate a fast agenda.
- void **e_activate_motors** (void(∗func1)(void), void(∗func2)(void))
- int **e_set_motor_speed** (void(∗func)(void), unsigned cycle)
- int **e_destroy_agenda** (void(∗func)(void))
  Destroy an agenda.
- int **e_set_agenda_cycle** (void(∗func)(void), unsigned cycle)
- int **e_reset_agenda** (void(∗func)(void))
  Reset an agenda's counter.

- int **e_pause_agenda** (void(∗func)(void))

  Pause an agenda.

- int **e_restart_agenda** (void(∗func)(void))

  Restart an agenda previously paused.

9.73.1 Detailed Description

Manage the fast agendas (timer1, 2, 3).

This module manage the fast agendas with the timer1, 2, 3.

An agenda is a structure made to work as chained list. It containts: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each times the timer1, 2, 3 has an interrupt, the corresponding agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, his callback function is launch.

Author:

Code: Julien Hubert
Doc: Jonathan Besuchet

9.73.2 Define Documentation

9.73.2.1 #define AG_ALREADY_CREATED 1

9.73.2.2 #define AG_NOT_FOUND 2

9.73.3 Typedef Documentation

9.73.3.1 typedef struct AgendaType Agenda

9.73.4 Function Documentation

9.73.4.1 void e_activate_agenda (void(∗)(void) *func*, unsigned *cycle*)

Activate a fast agenda.

Activate an agenda and allocate memory for him if there isn't already an agenda with the same callback function (the agenda is active but isn't processed if he have a null cycle value).

The appropriate timer is automatically selected.

Parameters:

func function called if the cycle value is reached by the counter

cycle cycle value in millisec/10

9.73.4.2 void e_activate_motors (void(∗)(void) *func1*, void(∗)(void) *func2*)

9.73.4.3 void e_configure_timer (int *timer*)

Configure the timer(s) used.

Configure one or all the timers to be used by the agenda

Parameters:

timer the timer's number to configure (between [0-3] with 0 configuring all of them)

9.73.4.4 int e_destroy_agenda (void(∗)(void) *func*)

Destroy an agenda.

Destroy the agenda with a given callback function.

Parameters:

func function to test

Returns:

EXIT_OK (p. 314) if the agenda has been destroyed, **AG_NOT_FOUND** (p. 319) otherwise

Destroy the agenda with a given callback function

Parameters:

func function to test

Returns:

int return the success of the destruction (EXIT_OK for successfull, AG_NOT_FOUND for unsuccessfull).

9.73.4.5 void e_end_agendas_processing (int *timer*)

Stop an agenda running.

Stop the agendas running on one particular timer (the memory allocated for the agenda isn't freed, use e_destroy_agenda for that).

Parameters:

timer the timer's number [0-3]. 0 stops all the timers.

See also:

e_destroy_agenda (p. 320)

9.73.4.6 int e_pause_agenda (void(∗)(void) *func*)

Pause an agenda.

Pause an agenda but do not reset its information.

Parameters:

func function to pause

Returns:

EXIT_OK (p. 314) the agenda has been paused, **AG_NOT_FOUND** (p. 319) otherwise

Pause an agenda but do not reset its information.

Parameters:

func function to pause

9.73.4.7 int e_reset_agenda (void(∗)(void) *func*)

Reset an agenda's counter.

Reset an agenda's counter with a given callback function.

Parameters:

func function to reset

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been reseted, **AG_NOT_FOUND** (p. 319) otherwise

Warning:

This function RESET the agenda, if you just want a pause tell **e_pause_agenda** (p. 321)(void (∗func)(void))

See also:

e_pause_agenda (p. 321)

Reset an agenda's counter.

Reset an agenda's counter with a given callback function

Parameters:

func function to reset

9.73.4.8 int e_restart_agenda (void(∗)(void) *func*)

Restart an agenda previously paused.

Restart an agenda previously paused.

Parameters:

func function to restart

Returns:

EXIT_OK (p. 314) if he agenda has been restarted, **AG_NOT_FOUND** (p. 319) otherwise

See also:

e_pause_agenda (p. 321)

Restart an agenda previously paused

Parameters:

func function to restart

9.73.4.9 int e_set_agenda_cycle (void(∗)(void) *func*, unsigned *cycle*)

9.73.4.10 int e_set_motor_speed (void(∗)(void) *func*, unsigned *cycle*)

9.73.4.11 void e_start_agendas_processing (void)

Start the agendas processing.

Start the agendas processing by starting the Timer2.

Start the agendas processing.

Start the agendas processing by initialising the accounting structures.

Don't activate any timer which is done by e_start_timer_processing.

See also:

e_start_timer_processing (p. 322)

9.73.4.12 void e_start_timer_processing (int *timer*)

Start the timer(s) used.

Activate one or all the timers to be used by the agenda.

Parameters:

timer the timer's number to activate (between [0-3] with 0 activating all of them)

9.74 motor_led/e_epuck_ports.h File Reference

Define all the usefull names corresponding of e-puck's hardware.

```
#include "p30f6014A.h"
```

Defines

- #define **FOSC** 7.3728e6
- #define **PLL** 8.0
- #define **FCY** ((FOSC∗PLL)/(4.0))
- #define **MILLISEC** (FCY/1.0e3)
- #define **MICROSEC** (FCY/1.0e6)
- #define **NANOSEC** (FCY/1.0e9)
- #define **TCY_PIC** (1e9/FCY)
- #define **INTERRUPT_DELAY** (10∗TCY_PIC)
- #define **TRUE** 1
- #define **FALSE** 0
- #define **OUTPUT_PIN** 0
- #define **LED0** _LATA6
- #define **LED1** _LATA7
- #define **LED2** _LATA9
- #define **LED3** _LATA12
- #define **LED4** _LATA10
- #define **LED5** _LATA13
- #define **LED6** _LATA14
- #define **LED7** _LATA15
- #define **LED0_DIR** _TRISA6
- #define **LED1_DIR** _TRISA7
- #define **LED2_DIR** _TRISA9
- #define **LED3_DIR** _TRISA12
- #define **LED4_DIR** _TRISA10
- #define **LED5_DIR** _TRISA13
- #define **LED6_DIR** _TRISA14
- #define **LED7_DIR** _TRISA15
- #define **FRONT_LED** _LATC1
- #define **FRONT_LED_DIR** _TRISC1
- #define **BODY_LED** _LATC2
- #define **BODY_LED_DIR** _TRISC2
- #define **PULSE_IR0** _LATF7
- #define **PULSE_IR1** _LATF8
- #define **PULSE_IR2** _LATG0
- #define **PULSE_IR3** _LATG1
- #define **PULSE_IR0_DIR** _TRISF7
- #define **PULSE_IR1_DIR** _TRISF8
- #define **PULSE_IR2_DIR** _TRISG0
- #define **PULSE_IR3_DIR** _TRISG1
- #define **IR0** 8
- #define **IR1** 9
- #define **IR2** 10

- #define **IR3** 11
- #define **IR4** 12
- #define **IR5** 13
- #define **IR6** 14
- #define **IR7** 15
- #define **MIC1** 2
- #define **MIC2** 3
- #define **MIC3** 4
- #define **ACCX** 5
- #define **ACCY** 6
- #define **ACCZ** 7
- #define **AUDIO_ON** _LATF0
- #define **AUDIO_ON_DIR** _TRISF0
- #define **MOTOR1_PHA** _LATD0
- #define **MOTOR1_PHB** _LATD1
- #define **MOTOR1_PHC** _LATD2
- #define **MOTOR1_PHD** _LATD3
- #define **MOTOR2_PHA** _LATD4
- #define **MOTOR2_PHB** _LATD5
- #define **MOTOR2_PHC** _LATD6
- #define **MOTOR2_PHD** _LATD7
- #define **MOTOR1_PHA_DIR** _TRISD0
- #define **MOTOR1_PHB_DIR** _TRISD1
- #define **MOTOR1_PHC_DIR** _TRISD2
- #define **MOTOR1_PHD_DIR** _TRISD3
- #define **MOTOR2_PHA_DIR** _TRISD4
- #define **MOTOR2_PHB_DIR** _TRISD5
- #define **MOTOR2_PHC_DIR** _TRISD6
- #define **MOTOR2_PHD_DIR** _TRISD7
- #define **CAM_RESET** _LATC13
- #define **CAM_RESET_DIR** _TRISC13
- #define **SIO_D** _LATG3
- #define **SIO_D_DIR** _TRISG3
- #define **SIO_C** _LATG2
- #define **SIO_C_DIR** _TRISG2
- #define **INPUT_PIN** 1
- #define **BATT_LOW** _RF1
- #define **BATT_LOW_DIR** _TRISF1
- #define **SELECTOR0** _RG6
- #define **SELECTOR1** _RG7
- #define **SELECTOR2** _RG8
- #define **SELECTOR3** _RG9
- #define **SELECTOR0_DIR** _TRISG6
- #define **SELECTOR1_DIR** _TRISG7
- #define **SELECTOR2_DIR** _TRISG8
- #define **SELECTOR3_DIR** _TRISG9
- #define **REMOTE** _RF6
- #define **REMOTE_DIR** _TRISF6
- #define **CAM_DATA** PORTD;
- #define **CAM_y0** _RD8

- #define **CAM_y1** _RD9
- #define **CAM_y2** _RD10
- #define **CAM_y3** _RD11
- #define **CAM_y4** _RD12
- #define **CAM_y5** _RD13
- #define **CAM_y6** _RD14
- #define **CAM_y7** _RD15
- #define **CAM_y0_DIR** _TRISD8
- #define **CAM_y1_DIR** _TRISD9
- #define **CAM_y2_DIR** _TRISD10
- #define **CAM_y3_DIR** _TRISD11
- #define **CAM_y4_DIR** _TRISD12
- #define **CAM_y5_DIR** _TRISD13
- #define **CAM_y6_DIR** _TRISD14
- #define **CAM_y7_DIR** _TRISD15
- #define **CAM_PWDN** _RC2
- #define **CAM_VSYNC** _RC4
- #define **CAM_HREF** _RC3
- #define **CAM_PCLK** _RC14
- #define **CAM_PWDN_DIR** _TRISC2
- #define **CAM_VSYNC_DIR** _TRISC4
- #define **CAM_HREF_DIR** _TRISC3
- #define **CAM_PCLK_DIR** _TRISC14
- #define **NOP**() {__asm__ volatile ("nop");}
- #define **CLRWDT**() {__asm__ volatile ("clrwdt");}
- #define **SLEEP**() {__asm__ volatile ("pwrsav #0");}
- #define **IDLE**() {__asm__ volatile ("pwrsav #1");}
- #define **INTERRUPT_OFF**() {__asm__ volatile ("disi #10000");}
- #define **INTERRUPT_ON**() {__asm__ volatile ("disi #2");}
- #define **RESET**() {__asm__ volatile ("reset");}
- #define **STOP_TMR1** IEC0bits.T1IE = 0
- #define **STOP_TMR2** IEC0bits.T2IE = 0
- #define **STOP_TMR3** IEC0bits.T3IE = 0
- #define **STOP_TMR4** IEC1bits.T4IE = 0
- #define **STOP_TMR5** IEC1bits.T5IE = 0

9.74.1 Detailed Description

Define all the usefull names corresponding of e-puck's hardware.

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.74.2 Define Documentation

9.74.2.1 #define ACCX 5

9.74.2.2 #define ACCY 6

9.74.2.3 #define ACCZ 7

9.74.2.4 #define AUDIO_ON _LATF0

9.74.2.5 #define AUDIO_ON_DIR _TRISF0

9.74.2.6 #define BATT_LOW _RF1

9.74.2.7 #define BATT_LOW_DIR _TRISF1

9.74.2.8 #define BODY_LED _LATC2

9.74.2.9 #define BODY_LED_DIR _TRISC2

9.74.2.10 #define CAM_DATA PORTD;

9.74.2.11 #define CAM_HREF _RC3

9.74.2.12 #define CAM_HREF_DIR _TRISC3

9.74.2.13 #define CAM_PCLK _RC14

9.74.2.14 #define CAM_PCLK_DIR _TRISC14

9.74.2.15 #define CAM_PWDN _RC2

9.74.2.16 #define CAM_PWDN_DIR _TRISC2

9.74.2.17 #define CAM_RESET _LATC13

9.74.2.18 #define CAM_RESET_DIR _TRISC13

9.74.2.19 #define CAM_VSYNC _RC4

9.74.2.20 #define CAM_VSYNC_DIR _TRISC4

9.74.2.21 #define CAM_y0 _RD8

9.74.2.22 #define CAM_y0_DIR _TRISD8

9.74.2.23 #define CAM_y1 _RD9

9.74.2.24 #define CAM_y1_DIR _TRISD9

9.74.2.25 #define CAM_y2 _RD10

9.74.2.26 #define CAM_y2_DIR _TRISD10

9.74.2.27 #define CAM_y3 _RD11

9.74.2.28 #define CAM_y3_DIR _TRISD11

9.74.2.29 #define CAM_y4 _RD12

9.74.2.30 #define CAM_y4_DIR _TRISD12

9.75 motor_led/e_init_port.c File Reference

Initialize the ports on standard configuration.

```
#include "e_epuck_ports.h"
```

Functions

- **_FOSC** (CSW_FSCM_OFF &XT_PLL8)
- **_FWDT** (WDT_OFF)
- **_FBORPOR** (PBOR_OFF &MCLR_EN)
- **_FGS** (CODE_PROT_OFF)
- void **e_init_port** (void)

  Initialize all ports (in/out).

9.75.1 Detailed Description

Initialize the ports on standard configuration.

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.75.2 Function Documentation

9.75.2.1 _FBORPOR (PBOR_OFF & *MCLR_EN*)

9.75.2.2 _FGS (CODE_PROT_OFF)

9.75.2.3 _FOSC (CSW_FSCM_OFF & *XT_PLL8*)

9.75.2.4 _FWDT (WDT_OFF)

9.75.2.5 void e_init_port (void)

Initialize all ports (in/out).

Call this method to set all the standards output components (LEDs, IR, camera, motors, I2C, audio) on their defaults values and set their corresponding PIN to "output". The method also set the corresponding PIN to "input" for all the standards inputs components (IR receiver, selector, camera, battery level).

9.76 motor_led/e_init_port.h File Reference

Initialize the ports on standard configuration.

Functions

- void **e_init_port** (void)

  Initialize all ports (in/out).

9.76.1 Detailed Description

Initialize the ports on standard configuration.

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.76.2 Function Documentation

9.76.2.1 void e_init_port (void)

Initialize all ports (in/out).

Call this method to set all the standards output components (LEDs, IR, camera, motors, I2C, audio) on their defaults values and set their corresponding PIN to "output". The method also set the corresponding PIN to "input" for all the standards inputs components (IR receiver, selector, camera, battery level).

9.77 motor_led/e_motors_timer3.c File Reference

Initialize the ports on standard configuration.

```
#include <stdlib.h>
#include "e_epuck_ports.h"
#include "e_init_port.h"
```

Functions

- void **__attribute__** ((interrupt, auto_psv, shadow))
- int **e_get_steps_left** (void)

  Give the number of left motor steps.

- void **e_set_steps_left** (int set_steps)

  Set the number of left motor steps.

- int **e_get_steps_right** (void)

  Give the number of right motor steps.

- void **e_set_steps_right** (int set_steps)

  Set the number of right motor steps.

- void **e_set_speed_left** (int motor_speed)

  Manage the left speed.

- void **e_set_speed_right** (int motor_speed)

  Manage the right speed.

- void **e_init_motors** (void)

  Initialize the motors's ports.

Variables

- static int **left_speed** = 0
- static int **right_speed** = 0
- static int **nbr_pas_left** = 0
- static int **nbr_pas_right** = 0
- static int **motor_counter_left** = 0
- static int **motor_counter_right** = 0
- static int **motor_counter_left_init** = 0
- static int **motor_counter_right_init** = 0

9.77.1 Detailed Description

Initialize the ports on standard configuration.

Manage the motors (with timer3).

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

This module manage the two motors with one timer: timer3.

Author:

Code: Michael Bonani, Francesco Mondada, Lucas Meier, Xavier Raemy
Doc: Jonathan Besuchet

9.77.2 Function Documentation

9.77.2.1 void __attribute__ ((interrupt, auto_psv, shadow))

9.77.2.2 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.77.2.3 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.77.2.4 void e_init_motors (void)

Initialize the motors's ports.

Initialize the motors's agendas.

This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

9.77.2.5 void e_set_speed_left (int *motor_speed*)

Manage the left speed.

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.77.2.6 void e_set_speed_right (int *motor_speed*)

Manage the right speed.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the timer3.

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.77.2.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.77.2.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

set_steps The number of changed phases that you want set.

9.77.3 Variable Documentation

9.77.3.1 int left_speed = 0 [static]

9.77.3.2 int motor_counter_left = 0 [static]

9.77.3.3 int motor_counter_left_init = 0 [static]

9.77.3.4 int motor_counter_right = 0 [static]

9.77.3.5 int motor_counter_right_init = 0 [static]

9.77.3.6 int nbr_pas_left = 0 [static]

9.77.3.7 int nbr_pas_right = 0 [static]

9.77.3.8 int right_speed = 0 [static]

9.78 uart/e_epuck_ports.inc File Reference

9.79 uart/e_uart_char.h File Reference

Manage UART.

Functions

- void **e_init_uart1** (void)

  Init uart 1 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.

- int **e_ischar_uart1** ()

  Check if something is comming on uart 1.

- int **e_getchar_uart1** (char ∗car)

  If available, read 1 char and put it in pointer.

- void **e_send_uart1_char** (const char ∗buff, int length)

  Send a buffer of char of size length.

- int **e_uart1_sending** (void)

  To check if the sending operation is done.

- void **e_init_uart2** (void)

  Init uart 2 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.

- int **e_ischar_uart2** ()

  Check if something is comming on uart 2.

- int **e_getchar_uart2** (char ∗car)

  If available, read 1 char and put it in pointer.

- void **e_send_uart2_char** (const char ∗buff, int length)

  Send a buffer of char of size length.

- int **e_uart2_sending** (void)

  To check if the sending operation is done.

Variables

- void ∗ **e_uart1_int_clr_addr**
- int **e_uart1_int_clr_mask**
- void ∗ **e_uart2_int_clr_addr**
- int **e_uart2_int_clr_mask**

9.79.1 Detailed Description

Manage UART.

This module manage all the UART ressource.

The e-puck's microcontroller has two integreted UART: UART1 and UART2.

A little exemple to comunicate with the e-puck through the uart. For this exemple you have to connect your e-puck to your PC with bluetooth (if you don't know how it works, look at the end of page 3 of this doc: `http://moodle.epfl.ch/mod/resource/view.php?id=12851`). Then open the HyperTerminal with the correct port COM and launch the connection. "Give a character:" should appears on the HyperTerminal.

```
#include <motor_led/e_init_port.h>
#include <uart/e_uart_char.h>

int main(void)
{
	char car;
	e_init_port();
	e_init_uart1();
	e_send_uart1_char("\f\a", 2);		//new page on HyperTerminal
	while(1)
	{
		e_send_uart1_char("Give a character:\r\n", 19);
		// do nothing while the text is not sent and while nothing is comming from the user
		while(e_uart1_sending() || !e_ischar_uart1()) {}
		e_getchar_uart1(&car);			// read the character entered...
		e_send_uart1_char("You have wrote: ", 16);
		e_send_uart1_char(&car, 1);		//... and resend him to uart.
		e_send_uart1_char("\r\n\r\n",4);
	}
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.79.2 Function Documentation

9.79.2.1 int e_getchar_uart1 (char * *car*)

If available, read 1 char and put it in pointer.

Parameters:

car The pointer where the caracter will be stored if available

Returns:

1 if a char has been readed, 0 if no char is available

9.79.2.2 int e_getchar_uart2 (char * *car*)

If available, read 1 char and put it in pointer.

Parameters:

car The pointer where the caracter will be stored if available

Returns:

1 if a char has been readed, 0 if no char is available

9.79.2.3 void e_init_uart1 (void)

Init uart 1 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.

9.79.2.4 void e_init_uart2 (void)

Init uart 2 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.

9.79.2.5 int e_ischar_uart1 ()

Check if something is comming on uart 1.

Returns:

the number of characters available, 0 if none are available

9.79.2.6 int e_ischar_uart2 ()

Check if something is comming on uart 2.

Returns:

the number of characters available, 0 if none are available

9.79.2.7 void e_send_uart1_char (const char * *buff*, int *length*)

Send a buffer of char of size length.

Parameters:

buff The top of the array where the data are stored

length The length of the array to send

9.79.2.8 void e_send_uart2_char (const char * *buff*, int *length*)

Send a buffer of char of size length.

Parameters:

buff The top of the array where the datas are stored

length The length of the array

9.79.2.9 int e_uart1_sending (void)

To check if the sending operation is done.

Returns:

1 if buffer sending is in progress, return 0 if not

9.79.2.10 int e_uart2_sending (void)

To check if the sending operation is done.

Returns:

1 if buffer sending is in progress, return 0 if not

9.79.3 Variable Documentation

9.79.3.1 void* e_uart1_int_clr_addr

9.79.3.2 int e_uart1_int_clr_mask

9.79.3.3 void* e_uart2_int_clr_addr

9.79.3.4 int e_uart2_int_clr_mask

Index