Program Listing for the e-puck Robot

This program listing is provided for easy reference and searchability. Indentations have not been preserved in the process of automatic inclusion in this listing file, and long lines of text may run off the edge of the page.

Contents

1 Demo Programs

1.1 Interactive Testing of Extended e-puck Functions: program/NUtest/NUtest.c

This program allows the user to test a number of functions of the e-puck using the e-puck library provided by EPFL, plus a few other functions for motor control created at NU in library/motor_led/advance_one_timer. Follow the instructions in the comments at the beginning of the program to create an MPLAB project and connect to the e-puck by hyperterminal.

```
/*********************************************************************************

  NUtest.c

  Allows the user to send commands and receive data from the e-puck via a simple
  hyperterminal interface.

  Created by Kevin Lynch, Dec 2007.
  Modified from the e-puck program main_com.c that works with the e-puck monitor program,
  to test most of the features of the e-puck.  Added some features, added some documentation,
  and simplified the code.

*********************************************************************************

Getting started:

(1) Create a folder where you will store this project under the folder e-puck\programs.
For example, you might call your folder e-puck\programs\NUtest.  Put this file NUtest.c
in this folder.

(2) Start MPLAB and create a new project using the MPLAB Project/Project Wizard.  The e-puck uses
the dsPIC30F6014A.  Choose the Microchip C30 Toolsuite.  Choose the folder you just created for your
project, and name your new project NUtest (MPLAB will add ".mcp" to the end of the name, so you
will now have a file e-puck\programs\NUtest\NUtest.mcp).  Now add the following files to
your project:

- This file (the main program), NUtest.c.
- C:\Program Files\Microchip\MPLAB C30\support\gld\p30f6014A.gld
- Then all the supporting .c function files, .h header files, .s assembly files, and .inc files,
  in the following directories in the e-puck library C:\e-puck\library:

  a_d/advance_ad_scan:   e_acc.{c,h}, e_ad_conv.{c,h}, e_micro.{c,h}, e_prox.{c,h}
  camera/fast_2_timer:   e_calc.c, e_interrupt.s, e_po3030k.h, e_registers.c, e_timers.c
  codec:                 e_common.inc, e_const_sound.s, e_init_codec_slave.s, e_init_dci_master.s,
                         e_isr_dci.s, e_sound.{c,h}, e_sub_dci_kickoff.s
  I2C (used by camera):  e_I2C_master_module.{c,h}, e_I2C_protocol.{c,h}
  motor_led:             e_epuck_ports.h, e_init_port.{c,h}
  motor_led/advance_one_timer:  e_agenda_NU.{c,h}, e_led.{c,h}, e_motors_NU.{c,h}
  uart:                  e_epuck_ports.inc, e_init_uart1.s, e_uart1_rx_char.s, e_uart1_tx_char.s,
                         e_uart_char.h

All of these are in the standard e-puck library distribution from EPFL, except for the files
e_agenda_NU.{c,h} and e_motors_NU.{c,h}, in motor_led/advance_one_timer.
These files are modifications of the original files to add a few features (like
dead reckoning of robot's configuration) and change the interval between motor interrupts
from 100 microseconds to 500 microseconds.

(3) After finishing with the Project Wizard, go to "Configure/Configuration Bits..." Uncheck
"Configuration Bits set in code" and then change the Oscillator to XT w/PLL 8x and the
Watchdog Timer to Disabled.

(4) Go to "Project/Build Options/Project" and make the following changes:

- Under "Directories," change "Intermediary directory" to ".\obj" by clicking "New" and
typing this in (not the quotes!)  This will cause all object files to be kept in this
directory, which is in the same directory as the project file.  This will keep all the
compiled object files from cluttering up the home directory.  Then change the "Assembler
Include Search Path" by adding two directories:  \library\codec and \libary\uart (add
them by "browsing" to them).  This tells MPLAB where to find the assembly .s files we need.
Then add the directory \library to the "Include Search Path," again by browsing to the
directory.  Finally, the "Library Search Path" should already have the Microchip Library present.

- Under MPLAB LINK30:  Set the Heap Size to 512 bytes.

- Under MPLAB C30:  Go to Categories: Memory Model and change the Data Model to Large data
model (it appears we have just enough variables to make this necessary).

(5) Go to "Project/Make" (or hit F10) to compile the program.  Ignore any warnings.
If compilation is successful, a file NUtest.hex is created in the project home directory.
This is the code that is actually loaded on the PIC.

(6) Make sure your e-puck is turned on and you have established a bluetooth connection.  Start
up the "Tiny Bootloader" program tinybldwin.exe.  Browse to your .hex file.  Make sure your
comm speed is set to 115200 and the port is set to the port on which you have a bluetooth
connection.  Click on Write Flash, then reset your e-puck by pressing the blue button.  The
bootloader should now find the e-puck and download your program.

(7) Start a hyperterminal session (Programs/Accessories/Communications) with the proper comm
settings (port number, speed 115200, data bits 8, parity none, stop bits 1) and begin
communicating with your e-puck!  It might not recognize your first couple of keystrokes, but
should after that.  Type h+enter for the menu of commands.  You might want to go to
File/Properties/Settings/ASCII Setup on hyperterminal to make it echo the characters you
type.

*********************************************************************************/

#include <p30f6014A.h>  // contains register definitions, etc., for the dsPIC

#include <string.h>  // string manipulation
#include <ctype.h>   // contains the "toupper" command to convert lower to uppercase
#include <stdio.h>   // standard I/O routines
#include <math.h>    // math functions like sqrt, cos, etc.
```

```
#include <a_d/advance_ad_scan/e_ad_conv.h>  // reads the A/D channels
#include <a_d/advance_ad_scan/e_acc.h>    // reads the 3 accelerometer channels (using e_ad_conv)
#include <a_d/advance_ad_scan/e_prox.h>    // reads the proximity sensors (using e_ad_conv)
#include <a_d/advance_ad_scan/e_micro.h>    // reads the 3 microphone volumes (using e_ad_conv)
#include <camera/fast_2_timer/e_po3030k.h>  // sets camera params, reads images to memory
#include <codec/e_sound.h>                  // speaker
#include <motor_led/e_epuck_ports.h>        // gives mnemonic names to pins, defines simple asm functions
#include <motor_led/e_init_port.h>          // initializes the port values and whether input or output
#include <motor_led/advance_one_timer/e_led.h>  // uses the LED's with interrupts, to allow blinking, etc.
#include <motor_led/advance_one_timer/e_motors_NU.h>  // interrupt-driven motor routines (step every x millisecs, e.g.)
#include <motor_led/advance_one_timer/e_agenda_NU.h>  // manages "agenda" (the interrupt routines)
#include <uart/e_uart_char.h>               // uses the uarts for comm (in this case, bluetooth with PC)

#define uart_send_static_text(msg) do { e_send_uart1_char(msg,sizeof(msg)-1); while(e_uart1_sending()); } while(0)
#define uart_send_text(msg) do { e_send_uart1_char(msg,strlen(msg)); while(e_uart1_sending()); } while(0)

// if SKIPROW is defined as 1, then prints images as 20x40 instead of 40x40; more square, fits on screen
#define SKIPROW 1

static char buffer[52*39*2+3+80];
extern int e_mic_scan[3][MIC_SAMP_NB];
extern unsigned int e_last_mic_scan_id;

// a function to return the absolute value of an integer
int intabs(int val) {
  if (val<0)
    return -val;
  else return val;
}

// a function to return the sign of an integer
int intsign(int val) {
  if (val < 0)
    return -1;
  else return 1;
}

// return the angle limited to [-PI, PI]; PI defined in e_motors_NU.h
float anglelimit(float ang) {

  while (ang<-PI) ang += 2.0*PI;
  while (ang>PI) ang -= 2.0*PI;
  return ang;
}

/* The main function of the program */
int main(void) {

  char c;
  int i,j,counter,speedr,speedl,LED_nbr,LED_action,accx,accy,accz,selector,sound;
  int circspeed,stepsl,stepsr;
  int cam_mode,cam_width,cam_height,cam_zoom,cam_size;
  static char first=0;
  int radius,rotationsense,threshold;
  float x,y,theta,dist,heading,diff,xgoal,ygoal,thetagoal,speed;
  char blackstring[2],whitestring[2];

  e_init_port();                  // configure port pins
  e_start_agendas_processing();   // start the motor interrupt service routines
  e_init_motors();
  e_init_uart1();                 // initialize UART to 115200 Kbaud
  e_init_ad_scan(ALL_ADC);

  if(RCONbits.POR) { // reset if power on (a problem for some robots).
    RCONbits.POR=0;     // RCONbits is defined in the p30f6014a.h file.
    RESET();
  }

  /* Camera default parameters */
  cam_width=40;
  cam_height=40;
  cam_zoom=8;
  threshold = 50;
  // default is grey scale; otherwise, use RGB_565_MODE and multiply cam_size by 2
  cam_mode=GREY_SCALE_MODE;
  cam_size=cam_width*cam_height;
  e_po3030k_init_cam();   // functions in camera/fast_2_timer/e_po3030k.h
  e_po3030k_config_cam((ARRAY_WIDTH -cam_width*cam_zoom)/2,(ARRAY_HEIGHT-cam_height*cam_zoom)/2,cam_width*cam_zoom,cam_height*cam_zoom,cam_zoom,cam_zoom,cam_mode);
  e_po3030k_set_mirror(1,1);
  e_po3030k_write_cam_registers();
  sprintf(blackstring,"%c",219);   // solid black for BW images, instead of just using "*"
  sprintf(whitestring," ");        // solid white for BW images, assumes background is white

  e_acc_calibr();
  e_calibrate_ir();

  uart_send_static_text("\f\a"
               "e-puck serial interface.  Type \"H\" for help. \r\n");

  while(1) {
    while (e_getchar_uart1(&c)==0)      // function in uart/e_uart_char.h
      if (c>0) {   // character received in ascii mode
        while (c=='\n' || c=='\r')
          e_getchar_uart1(&c);          // keep reading characters until not a newline or return
        buffer[0]=c;                    // put first character in the string

        i = 1;
        do if (e_getchar_uart1(&c))    // "do" put chars in string while chars available
          buffer[i++]=c;
        while (c!='\n' && c!='\r');    // end "do" when char is newline or return

        buffer[i++]='\0';              // end the string
        buffer[0]=toupper(buffer[0]);  // convert lowercase to upper

        switch (buffer[0]) {
```

```
          case 'A':  // read accelerometer, functions in a_d/advance_ad_scan/e_acc.h
            accx=e_get_acc(0);
            accy=e_get_acc(1);
            accz=e_get_acc(2);
            sprintf(buffer,"A, xacc: %d, yacc: %d, zacc: %d\r\n",accx,accy,accz);
            uart_send_text(buffer);
            break;

          case 'B':  // set body LED, function in motor_led/advance_one_timer/e_led.h
            sscanf(buffer,"B,%d\r",&LED_action);
            e_set_body_led(LED_action);
            uart_send_static_text("b\r\n");
            break;

          case 'C':  // read selector position
            selector = SELECTOR0 + 2*SELECTOR1 + 4*SELECTOR2 + 8*SELECTOR3;
            sprintf(buffer,"C,%d\r\n",selector);
            uart_send_text(buffer);
            break;

          case 'D':  // set motor speed, functions in motor_led/advance_one_timer/e_motors_NU.h
            sscanf(buffer, "D,%d,%d\r", &speedl, &speedr);
            e_set_speed_left(speedl);
            e_set_speed_right(speedr);
            uart_send_static_text("d\r\n");
            break;

          case 'E':  // return the motor speeds
            sprintf(buffer,"E,%d,%d\r\n",speedl,speedr);
            uart_send_text(buffer);
            break;

          case 'F':  // set front LED, function in motor_led/advance_one_timer/e_led.h
            sscanf(buffer,"F,%d\r",&LED_action);
            e_set_front_led(LED_action);  //
            uart_send_static_text("f\r\n");
            break;

          case 'H':  // print command menu
            uart_send_static_text("\n");
            uart_send_static_text("\"A\"           Accelerometer\r\n");
            uart_send_static_text("\"B,#\"         Body led 0=off 1=on 2=inverse\r\n");
            uart_send_static_text("\"C\"           Selector position\r\n");
            uart_send_static_text("\"D,#,#\"       Set motor speed left,right\r\n");
            uart_send_static_text("\"E\"           Get motor speed left,right\r\n");
            uart_send_static_text("\"F,#\"         Front led 0=off 1=on 2=inverse\r\n");
            uart_send_static_text("\"H\"           Help\r\n");
            uart_send_static_text("\"I\"           Print camera parameters (threshold, mode) and image\r\n");
            uart_send_static_text("\"J,#,#\"       Set camera threshold (0-255), mode (0=BW, 1=RGB)\r\n");
            uart_send_static_text("\"K\"           Calibrate proximity sensors\r\n");
            uart_send_static_text("\"L,#,#\"       Led number,0=off 1=on 2=inverse\r\n");
            uart_send_static_text("\"N\"           Proximity\r\n");
            uart_send_static_text("\"O\"           Light sensors\r\n");
            uart_send_static_text("\"R\"           Reset e-puck\r\n");
            uart_send_static_text("\"S\"           Stop e-puck and turn off leds\r\n");
            uart_send_static_text("\"T,#\"         Play sound 1-5 else stop sound\r\n");
            uart_send_static_text("\"U\"           Get microphone amplitude\r\n");
            uart_send_static_text("\"V\"           Print robot's current configuration (in cm and degrees)\r\n");
            uart_send_static_text("\"W,#,#,#\"     Set configuration (in cm and degrees)\r\n");
            uart_send_static_text("\"X,#,#,#\"     Move to x, y, theta (cm and degrees)\r\n");
            uart_send_static_text("\"Y,#,#,#,#\"   Left wheel speed (-1000 to 1000), dist (in stepper counts), right speed, dist\r\n");
            uart_send_static_text("\"Z,#,#,#\"     Circle of radius (int cm), speed (1 to 1000), sense (0=CW, 1=CCW) \r\n");
            break;

          case 'I':  // print camera parameters and image
            sprintf(buffer,"I,%d,%d\r\n",threshold,cam_mode);
            uart_send_text(buffer);
    e_po3030k_launch_capture(&buffer[0]);
            while(!e_po3030k_is_img_ready());
    if(cam_mode!=GREY_SCALE_MODE) {
      e_send_uart1_char(buffer,cam_size);  // send raw RGB image.  warning:  prints as garbage!
      while(e_uart1_sending());
      uart_send_static_text("\r\n");
    }
    else {
              // Display thresholded BW image.
              // For example, hold a 5 cm x 5 cm X in black magic marker on white paper about
              // 8 cm in front of the camera, with threshold of 30 or 50 or so (adjust until you see the X).
              // Note moving target
              // to left (from the camera's view) moves it up in the display, and moving target down
              // (to camera) moves it left in display.
      counter=0;
      for(i=0; i<cam_height; i++) {
if ((i%2==1) && (SKIPROW==1)) counter += cam_width;  // jump ahead in image buffer to skip row
else {
  for(j=0; j<cam_width; j++) {
    if (buffer[counter]<threshold) uart_send_text(blackstring);
    else uart_send_text(whitestring);
    counter++;
  }
  uart_send_static_text("\r\n");
}
      }
    }
            break;

          case 'J':  //set camera parameters; see also camera/fast_2_timer/e_calc.c for other settings
            sscanf(buffer,"J,%d,%d\r",&threshold,&cam_mode);
            if(cam_mode==GREY_SCALE_MODE)  // GREY_SCALE_MODE = 0 in e_po3030k.h; RGB_565_MODE = 1
              cam_size=cam_width*cam_height;
            else {
              cam_size=cam_width*cam_height*2;
      cam_mode=RGB_565_MODE;
    }
            e_po3030k_init_cam();
            e_po3030k_config_cam((ARRAY_WIDTH -cam_width*cam_zoom)/2,(ARRAY_HEIGHT-cam_height*cam_zoom)/2,cam_width*cam_zoom,cam_height*cam_zoom,cam_zoom,cam_zoom,cam_mode);
            e_po3030k_set_mirror(1,1);
```

```
        e_po3030k_write_cam_registers();
uart_send_text(buffer);
        uart_send_static_text("\r\n");
        break;

      case 'K':  // calibrate proximity sensors, in a_d/advance_ad_scan/e_prox.h
        uart_send_static_text("K, Starting calibration - Remove any object in sensors' range\r\n");
        e_calibrate_ir();
        uart_send_static_text("K, Calibration finished\r\n");
        break;

      case 'L':  // set LED, see motor_led/advance_one_timer/e_led.h
        sscanf(buffer,"L,%d,%d\r",&LED_nbr,&LED_action);
        e_set_led(LED_nbr,LED_action);
        uart_send_static_text("l\r\n");
        break;

      case 'N':  // read proximity sensors, a_d/advance_ad_scan/e_prox.h
        sprintf(buffer,"N,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
        e_get_calibrated_prox(0),e_get_calibrated_prox(1),e_get_calibrated_prox(2),e_get_calibrated_prox(3),
        e_get_calibrated_prox(4),e_get_calibrated_prox(5),e_get_calibrated_prox(6),e_get_calibrated_prox(7));
        uart_send_text(buffer);
        break;

      case 'O':  // read ambient light (IR sensors), a_d/advance_ad_scan/e_prox.h
        sprintf(buffer,"O,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
        e_get_ambient_light(0),e_get_ambient_light(1),e_get_ambient_light(2),e_get_ambient_light(3),
        e_get_ambient_light(4),e_get_ambient_light(5),e_get_ambient_light(6),e_get_ambient_light(7));
        uart_send_text(buffer);
        break;

      case 'R':  // reset, function in motor_led/e_epuck_ports.h
        uart_send_static_text("R\r\n");
        RESET();
        break;

      case 'S':  // stop, motor_led/advance_one_timer/{e_motors_NU.h, e_led.h}
        e_set_speed_left(0);
        e_set_speed_right(0);
        e_goal_active_left(0);
        e_goal_active_right(0);
        e_set_led(8,0);
        uart_send_static_text("S\r\n");
        break;

      case 'T':  // use the speaker, codec/e_sound.h
        sscanf(buffer,"T,%d",&sound);
        if(first==0){
          e_init_sound();
          first=1;
        }
        switch(sound)
        {
          case 1: e_play_sound(0,2112);break;
          case 2: e_play_sound(2116,1760);break;
          case 3: e_play_sound(3878,3412);break;
          case 4: e_play_sound(7294,3732);break;
          case 5: e_play_sound(11028,8016);break;
          default:
            e_close_sound();
            first=0;
            break;
        }
uart_send_text(buffer);
        uart_send_static_text("\r\n");
        break;

      case 'U':  // check microphones, a_d/advance_ad_scan/micro.h
        sprintf(buffer,"U,%d,%d,%d\r\n",e_get_micro_volume(0),e_get_micro_volume(1),e_get_micro_volume(2));
        uart_send_text(buffer);
        break;

      case 'V':  // get current estimated config, motor_led/advance_one_timer/e_motors_NU.h
        e_get_configuration(&x,&y,&theta);
        sprintf(buffer,"V, Current configuration (cm, deg):  x: %6.2f   y: %6.2f   theta: %6.2f\r\n",x,y,theta*RAD2DEG);
        uart_send_text(buffer);
        break;

      case 'W':  // set the config, e.g., as if receiving GPS data, to correct any accumulated error
        sscanf(buffer,"W,%f,%f,%f\r",&x,&y,&theta);
        e_set_configuration(x,y,theta*DEG2RAD);
        e_get_configuration(&x,&y,&theta);
        sprintf(buffer,"Configuration set to (cm, deg):  x: %6.2f   y: %6.2f   theta: %6.2f\r\n",x,y,theta*RAD2DEG);
        uart_send_text(buffer);
        break;

      case 'X':  // move to a new config by rotate, translate, rotate (e_motors_NU.h)
        sscanf(buffer,"X,%f,%f,%f\r",&xgoal,&ygoal,&thetagoal);
        sprintf(buffer,"X, Moving to (%6.2f, %6.2f, %6.2f)\r\n",xgoal,ygoal,thetagoal);
        uart_send_text(buffer);
        thetagoal = anglelimit(thetagoal*DEG2RAD);
        e_get_configuration(&x,&y,&theta);
        dist = sqrt((x-xgoal)*(x-xgoal)+(y-ygoal)*(y-ygoal));
        if (dist>0.1) {  // don't do the first rotate and translate if goal is too close
          heading = atan2(ygoal-y,xgoal-x);
          diff = anglelimit(heading-theta);
          if (diff<(-0.5*PI)) {
            dist = -dist;
            diff = diff+PI;
          }
          else if (diff>(0.5*PI)) {
            dist = -dist;
            diff = diff-PI;
          }
          sprintf(buffer,"e_rotate(%f,3.0)\r\n",diff);
          uart_send_text(buffer);
          speed = e_rotate(diff,3.0);     // rotate at 3 radians/sec
```

```
            while((e_get_goal_active_left()!=0) || (e_get_goal_active_right()!=0));
            uart_send_static_text("Completed first rotation...\r\n");
            theta = theta + diff;
            sprintf(buffer,"e_translate(%f,8.0)\r\n",dist);
            uart_send_text(buffer);
            speed = e_translate(dist,8.0);  // translate at 8 cm/sec (max around 12.9 cm/sec)
            while((e_get_goal_active_left()!=0) || (e_get_goal_active_right()!=0));
            uart_send_static_text("completed translation...\r\n");
          }
          diff = anglelimit(thetagoal - theta);
          if (abs(diff)>0.02) {             // don't do final rotate if already close to desired angle
            sprintf(buffer,"e_rotate(%f,3.0)\r\n",diff);
            uart_send_text(buffer);
            speed = e_rotate(diff,3.0);     // rotate at 3 radians/sec
            while((e_get_goal_active_left()!=0) || (e_get_goal_active_right()!=0));
          }
          uart_send_static_text("completed move.\r\n");
          break;

        case 'Y':  // set wheel speeds and number of steps (e_motors_NU.h)
          sscanf(buffer,"Y,%d,%d,%d,%d\r",&speedl,&stepsl,&speedr,&stepsr);
          e_set_steps_left(0);        // set current left motor position (in angle steps)
          e_set_steps_right(0);       // set current right motor position
          e_goal_active_left(1);      // left motor will move to a goal position
          e_goal_active_right(1);     // right motor will move to a goal position
          e_goal_steps_left(stepsl);  // set the goal steps for left motor
          e_goal_steps_right(stepsr); // set the goal steps for right motor
          e_set_speed_left(speedl);   // set left motor speed
          e_set_speed_right(speedr);  // set right motor speed
          uart_send_text(buffer);
    uart_send_static_text("\r\n");
          break;

        case 'Z':  // added this to make it move in a circle
          sscanf(buffer,"Z,%d,%d,%d\r",&radius,&circspeed,&rotationsense);
          stepsr = (int) (STEPS_PER_REV*(radius-WHEELBASE)/WHEELRADIUS);
          stepsl = (int) (STEPS_PER_REV*(radius+WHEELBASE)/WHEELRADIUS);
          if (intabs(stepsr)>intabs(stepsl)) {
            speedr = intsign(stepsr)*circspeed;
            speedl = (int) (  (float) (stepsl) * (float) (speedr) /  (float) (stepsr));
          }
          else {
            speedl = intsign(stepsl)*circspeed;
            speedr = (int) (  (float) (stepsr) * (float) (speedl) /  (float) (stepsl));
          }
          if (rotationsense == 1) {
            speedr = -speedr; stepsr = -stepsr;
            speedl = -speedl; stepsl = -stepsl;
          }
          sprintf(buffer,"stepsr:  %d  stepsl:  %d  speedr:  %d  speedl  %d      \r\n",stepsr,stepsl,speedr,speedl);
          uart_send_text(buffer);
          e_set_steps_left(0);
          e_set_steps_right(0);
          e_goal_active_left(1);
          e_goal_active_right(1);
          e_goal_steps_left(stepsl);
          e_goal_steps_right(stepsr);
          e_set_speed_left(speedl);
          e_set_speed_right(speedr);
          break;

        default:
          uart_send_static_text("Command not found\r\n");
          break;
      }
    }
  }
}
```

1.2 Interactive Testing of e-puck Functions: program/advance_sercom/main_com.c

This program is provided by EPFL and can be used with the e-puck monitor program (in the "tool" directory) to interactively operate the e-puck with a GUI interface.

```
/***************************************************************************

The reference programm to control the e-puck with the PC
Version 2.0
Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \brief The main file of the programm.
 *
 * The sercom works as folowing. Into the infinite loop we look if something is
 * coming from the uart1. If a good command is coming we branch it to the correct
 * code and we send back at first the comand followed by the result (if there is one).
 * \n \n For exemple if the "a + return" is coming, we return first the command: "a"
 * and then the three values of the accelerometer separated by a coma, the returned
```

```
 * value will be for exemple: "a,2001,2300,1890+return"
 * \n \n Another exemple is controling the motors. In this case we receive for exemple
 * "d,200,-200+return". Then we put the motor speed left on 200 and motor speed right
 * on -200 and send back "d+return".
 * \warning To compile the programm, you MUST to set the define variable MIC_SAMP_NB
 * in the file library\a_d\advance_ad_scan\e_ad_con.h with 1000
 * \n It must look like this: #define MIC_SAMP_NB 100
 * \author Jonathan Besuchet
 */

/*! \mainpage Sercom programm
 * \section intro_sec Introduction
 * SerCom is a simple text-based protocol that allows the PC to control the e-puck. With SerCom, the PC can read
 * the e-puck's sensors and set the e-puck's actuators. The PC can be a human using a serial communication program
 * (such as Hyperterminal) or a program using a serial port library for exemple you can use the e-puck_monitor.exe
 * locate in "tool\e-puck_monitor".
 *
 * \section sect_run Compiling and running
 * First of all you have to compile the programm.
 * \n \n Flash the programm to the e-puck.
 * \warning To compile the programm, you MUST to set the define variable MIC_SAMP_NB
 * in the file library\a_d\advance_ad_scan\e_ad_con.h with 1000
 * \n It must look like this: #define MIC_SAMP_NB 100
 *
 * \section sect_hyperterminal Using sercom with Hyperterminal
 * Pair the e-puck with your computer: this will create a new
 * virtual COM port that you can use to communicate with the e-puck.
 * \n With Hyperterminal, connect to this COM port and play with the e-puck serial protocol. The available commands
 * are listed in this file: http://asl.epfl.ch/epfl/education/courses/MicroInfo2/TP/sercom1.02.pdf . You can also
 * press 'H' + return to see the list of available commands.
 *
 * \section link_sec External links
 * - http://www.e-puck.org/               The official site of the e-puck
 * - https://gna.org/projects/e-puck/     The developpers 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
 *
 * \author Code: Micahael Bonani \n Doc: Jonathan Besuchet
 */

#include <p30f6014A.h>

#define FLOOR_SENSORS // define to enable floor sensors
//#define LIS_SENSORS_TURRET
#define IR_RECIEVER

#include <string.h>
#include <ctype.h>
#include <stdio.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_motors.h>

#include <uart/e_uart_char.h>
#include <a_d/advance_ad_scan/e_ad_conv.h>
#include <a_d/advance_ad_scan/e_acc.h>
#include <a_d/advance_ad_scan/e_prox.h>
#include <a_d/advance_ad_scan/e_micro.h>
#include <motor_led/advance_one_timer/e_agenda.h>
#include <camera/fast_2_timer/e_po3030k.h>
#include <codec/e_sound.h>

#ifdef LIS_SENSORS_TURRET
//#include <contrib/LIS_sensors_turret/e_devantech.h>
#include <contrib/LIS_sensors_turret/e_sharp.h>
#include <contrib/LIS_sensors_turret/e_sensext.h>
#include <I2C/e_I2C_master_module.h>
#include <I2C/e_I2C_protocol.h>
#endif

#ifdef FLOOR_SENSORS
#include <./I2C/e_I2C_protocol.h>
#endif

#ifdef IR_RECIEVER
#include <motor_led/advance_one_timer/e_remote_control.h>
#define SPEED_IR 600
#endif

#define uart_send_static_text(msg) do { e_send_uart1_char(msg,sizeof(msg)-1); while(e_uart1_sending()); } while(0)
#define uart_send_text(msg) do { e_send_uart1_char(msg,strlen(msg)); while(e_uart1_sending()); } while(0)


static char buffer[52*39*2+3+80];

extern int e_mic_scan[3][MIC_SAMP_NB];
extern unsigned int e_last_mic_scan_id;


/* \brief The main function of the programm */
int main(void) {
char c,c1,c2,wait_cam=0;
int i,j,n,speedr,speed1,positionr,position1,LED_nbr,LED_action,accx,accy,accz,selector,sound;
int cam_mode,cam_width,cam_heigth,cam_zoom,cam_size;
static char first=0;
char *address;
char *ptr;
#ifdef LIS_SENSORS_TURRET
int sensext_pres=1, sensext_param[2], sensext_debug=0;
unsigned int sensext_value[2];
#endif
#ifdef IR_RECIEVER
char ir_move = 0,ir_address= 0, ir_last_move = 0;
#endif
TypeAccSpheric accelero;
```

```
e_init_port();    // configure port pins
e_start_agendas_processing();
e_init_motors();
e_init_uart1();   // initialize UART to 115200 Kbaud
e_init_ad_scan(ALL_ADC);
#ifdef FLOOR_SENSORS
#ifndef LIS_SENSORS_TURRET
e_i2cp_init();
#endif
#endif

#ifdef IR_RECIEVER
e_init_remote_control();
#endif
if(RCONbits.POR) { // reset if power on (some problem for few robots)
RCONbits.POR=0;
RESET();
}

/*Cam default parameter*/
cam_mode=RGB_565_MODE;
cam_width=40;
cam_heigth=40;
cam_zoom=8;
cam_size=cam_width*cam_heigth*2;
e_po3030k_init_cam();
e_po3030k_config_cam((ARRAY_WIDTH -cam_width*cam_zoom)/2,(ARRAY_HEIGHT-cam_heigth*cam_zoom)/2,cam_width*cam_zoom,cam_heigth*cam_zoom,cam_zoom,cam_zoom,cam_mode);
    e_po3030k_set_mirror(1,1);
    e_po3030k_write_cam_registers();

#ifdef LIS_SENSORS_TURRET //check if sensor extension is present and initalizes ports accordingly

e_i2cp_init();
e_i2cp_enable();
sensext_debug = e_i2cp_write (I2C_ADDR_SENSEXT , 0, 49);
e_i2cp_disable();

// Wait for I2C eeprom on tourret to answer.
e_activate_agenda(e_stop_sensext_wait,0);
e_start_sensext_wait();
e_set_agenda_cycle(e_stop_sensext_wait, 100);
while(e_get_sensext_wait());
e_set_agenda_cycle(e_stop_sensext_wait, 0);

e_i2cp_enable();
sensext_debug = e_i2cp_read(I2C_ADDR_SENSEXT, 0);
e_i2cp_disable();

if(sensext_debug!=49) // no SENSORS_TURRET reply
{
sensext_pres=0;

}
else
{
sensext_pres=1;
e_init_sensext();
e_init_sharp(); //a executer s'il y a la tourelle
// uart_send_static_text("ePic\r\n");
}
#endif


    e_acc_calibr();
e_calibrate_ir();

uart_send_static_text("\f\a"
                      "WELCOME to the SerCom protocol on e-Puck\r\n"
                      "the EPFL education robot type \"H\" for help\r\n");
while(1) {
while (e_getchar_uart1(&c)==0)
#ifdef IR_RECIEVER
{

ir_move = e_get_data();
ir_address = e_get_address();
if (((ir_address ==  0)||(ir_address ==  8))&&(ir_move!=ir_last_move)){
switch(ir_move)
{
case 1:
speedr = SPEED_IR;
speedl = SPEED_IR/2;
break;
case 2:
speedr = SPEED_IR;
speedl = SPEED_IR;
break;
case 3:
speedr = SPEED_IR/2;
speedl = SPEED_IR;
break;
case 4:
speedr = SPEED_IR;
speedl = -SPEED_IR;
break;
case 5:
speedr = 0;
speedl = 0;
break;
case 6:
speedr = -SPEED_IR;
speedl = SPEED_IR;
break;
case 7:
speedr = -SPEED_IR;
speedl = -SPEED_IR/2;
break;
```

```
case 8:
speedr = -SPEED_IR;
speedl = -SPEED_IR;
break;
case 9:
speedr = -SPEED_IR/2;
speedl = -SPEED_IR;
break;
case 0:
if(first==0){
e_init_sound();
first=1;
}
e_play_sound(11028,8016);
break;
default:
speedr = speedl = 0;
}
ir_last_move = ir_move;
e_set_speed_left(speedl);
e_set_speed_right(speedr);
}

}
#else
;
#endif
if (c<0) { // binary mode (big endian)
i=0;
do {
switch(-c) {
case 'a':  // Read acceleration sensors in a non filtered way, some as ASCII
accx=e_get_acc(0);
accy=e_get_acc(1);
accz=e_get_acc(2);
ptr=(char *)&accx;
buffer[i++]=accx & 0xff;
buffer[i++]=accx>>8;

buffer[i++]=accy & 0xff;
buffer[i++]=accy>>8;

buffer[i++]=accz & 0xff;
buffer[i++]=accz>>8;

break;
case 'A': // read acceleration sensors
accelero=e_read_acc_spheric();
ptr=(char *)&accelero.acceleration;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);

ptr=(char *)&accelero.orientation;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);

ptr=(char *)&accelero.inclination;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);
ptr++;
buffer[i++]=(*ptr);

break;
case 'D': // set motor speed
while (e_getchar_uart1(&c1)==0);
while (e_getchar_uart1(&c2)==0);
speedl=(unsigned char)c1+((unsigned int)c2<<8);
while (e_getchar_uart1(&c1)==0);
while (e_getchar_uart1(&c2)==0);
speedr=(unsigned char)c1+((unsigned  int)c2<<8);
e_set_speed_left(speedl);
e_set_speed_right(speedr);
break;
case 'E': // get motor speed
buffer[i++]=speedl & 0xff;
buffer[i++]=speedl >> 8;

buffer[i++]=speedr & 0xff;
buffer[i++]=speedr >> 8;

break;
case 'I': // get camera image
e_po3030k_launch_capture(&buffer[i+3]);
wait_cam=1;
buffer[i++]=(char)cam_mode&0xff;//send image parameter
buffer[i++]=(char)cam_width&0xff;
buffer[i++]=(char)cam_heigth&0xff;
i+=cam_size;
break;
case 'L': // set LED
while (e_getchar_uart1(&c1)==0);
while (e_getchar_uart1(&c2)==0);
switch(c1) {
case 8:
```

```
  e_set_body_led(c2);
  break;
case 9:
  e_set_front_led(c2);
  break;
default:
            e_set_led(c1,c2);
            break;
          }
break;
case 'M': // optional floor sensors
#ifdef FLOOR_SENSORS
e_i2cp_enable();

for (j=0; j<3; ++j) {
buffer[i++] = e_i2cp_read(0xC0,2*j+1);
buffer[i++] = e_i2cp_read(0xC0,2*j);
}

e_i2cp_disable();
#else
for(j=0;j<6;j++) buffer[i++]=0;
#endif
break;
case 'N': // read proximity sensors
for(j=0;j<8;j++) {
n=e_get_calibrated_prox(j);
buffer[i++]=n&0xff;
buffer[i++]=n>>8;
}
break;
case 'O': // read light sensors
for(j=0;j<8;j++) {
n=e_get_ambient_light(j);
buffer[i++]=n&0xff;
buffer[i++]=n>>8;
}
break;
case 'Q': // read encoders
                    n=e_get_steps_left();
buffer[i++]=n&0xff;
buffer[i++]=n>>8;
                    n=e_get_steps_right();
buffer[i++]=n&0xff;
buffer[i++]=n>>8;
break;
case 'u': // get last micro volumes
n=e_get_micro_volume(0);
buffer[i++]=n&0xff;
buffer[i++]=n>>8;

n=e_get_micro_volume(1);
buffer[i++]=n&0xff;
buffer[i++]=n>>8;

n=e_get_micro_volume(2);
buffer[i++]=n&0xff;
buffer[i++]=n>>8;
break;
case 'U': // get micro buffer
address=(char *)e_mic_scan;
e_send_uart1_char(address,600);//send sound buffer
n=e_last_mic_scan_id;//send last scan
buffer[i++]=n&0xff;
break;
case 'W': // read Devantec ultrasonic range sensor or Sharp Ir sensor (optional)
#ifdef LIS_SENSORS_TURRET
while (e_getchar_uart1(&c1)==0);
while (e_getchar_uart1(&c2)==0);
sensext_param[0] = (unsigned char)c1+((unsigned int)c2<<8);
// sscanf(buffer,"W,%d\r",&sensext_param[0]);
if (sensext_pres) // If the TP_sensors tourret is present
{
if(e_sensext_process(sensext_param, sensext_value))
{
switch (sensext_param[0])
{
case -1: //i2c SRFxxx
buffer[i++] = sensext_value[0]&0xff;
buffer[i++] = sensext_value[0]>>8;

buffer[i++] = sensext_value[1]&0xff;
buffer[i++] = sensext_value[1]>>8;
// sprintf(buffer,"w,%u,%u\r\n", sensext_value[0],  sensext_value[1]);
break;
case -2: // i2c cmps03
buffer[i++] = sensext_value[0]&0xff;
buffer[i++] = sensext_value[0]>>8;

buffer[i++] = sensext_value[1]&0xff;
buffer[i++] = sensext_value[1]>>8;
// sprintf(buffer,"w,%d,%d\r\n",  sensext_value[0],  sensext_value[1]);
break;
default: //analog (sharp,...)
buffer[i++] = (sensext_value[0]&0xff);
buffer[i++] = (sensext_value[0]>>8);
// sprintf(buffer,"w,%d\r\n",  sensext_value[0]);
break;
}
// uart_send_text(buffer);
}
else
{
switch(sensext_param[0])
{
case -1:
case -2:
```

```
sensext_value[0] = -1;
sensext_value[1] = -1;
buffer[i++] = sensext_value[0]&0xff;
buffer[i++] = sensext_value[0]>>8;

buffer[i++] = sensext_value[1]&0xff;
buffer[i++] = sensext_value[1]>>8;
break;
default:
sensext_value[0] = -1;
sensext_value[1] = -1;
buffer[i++] = sensext_value[0]&0xff;
buffer[i++] = sensext_value[0]>>8;
break;
}
// uart_send_static_text("wrong parameter\r\n");
}

}
else
{
uart_send_static_text("LIS sensors turret not present\r\n");
}
#endif
break;
default: // silently ignored
break;
}
while (e_getchar_uart1(&c)==0); // get next command
} while(c);
if (i!=0){
if (wait_cam) {
wait_cam=0;
while(!e_po3030k_is_img_ready());
}
e_send_uart1_char(buffer,i); // send answer
while(e_uart1_sending());
}
} else if (c>0) { // ascii mode
while (c=='\n' || c=='\r')
 e_getchar_uart1(&c);
buffer[0]=c;
i = 1;
do if (e_getchar_uart1(&c))
buffer[i++]=c;
while (c!='\n' && c!='\r');
buffer[i++]='\0';
buffer[0]=toupper(buffer[0]); // we also accept lowercase letters
switch (buffer[0]) {
case 'A': // read accelerometer
accx=e_get_acc(0);
accy=e_get_acc(1);
accz=e_get_acc(2);
sprintf(buffer,"a,%d,%d,%d\r\n",accx,accy,accz);
uart_send_text(buffer);
/* accelero=e_read_acc_spheric();
sprintf(buffer,"a,%f,%f,%f\r\n",accelero.acceleration,accelero.orientation,accelero.inclination);
uart_send_text(buffer);*/
break;
case 'B': // set body led
sscanf(buffer,"B,%d\r",&LED_action);
  e_set_body_led(LED_action);
uart_send_static_text("b\r\n");
break;
case 'C': // read selector position
selector = SELECTOR0 + 2*SELECTOR1 + 4*SELECTOR2 + 8*SELECTOR3;
sprintf(buffer,"c,%d\r\n",selector);
uart_send_text(buffer);
break;
case 'D': // set motor speed
sscanf(buffer, "D,%d,%d\r", &speedl, &speedr);
e_set_speed_left(speedl);
e_set_speed_right(speedr);
uart_send_static_text("d\r\n");
break;
case 'E': // read motor speed
sprintf(buffer,"e,%d,%d\r\n",speedl,speedr);
uart_send_text(buffer);
break;
case 'F': // set front led
sscanf(buffer,"F,%d\r",&LED_action);
e_set_front_led(LED_action);
uart_send_static_text("f\r\n");
break;
#ifdef IR_RECIEVER
case 'G':
                  sprintf(buffer,"g IR check : 0x%x, address : 0x%x, data : 0x%x\r\n", e_get_check(), e_get_address(), e_get_data());
                  uart_send_text(buffer);
                  break;
#endif
case 'H': // ask for help
uart_send_static_text("\n");
uart_send_static_text("\"A\"         Accelerometer\r\n");
uart_send_static_text("\"B,#\"       Body led 0=off 1=on 2=inverse\r\n");
uart_send_static_text("\"C\"         Selector position\r\n");
uart_send_static_text("\"D,#,#\"     Set motor speed left,right\r\n");
uart_send_static_text("\"E\"         Get motor speed left,right\r\n");
uart_send_static_text("\"F,#\"       Front led 0=off 1=on 2=inverse\r\n");
#ifdef IR_RECIEVER
uart_send_static_text("\"G\"         IR receiver\r\n");
#endif
uart_send_static_text("\"H\"      Help\r\n");
uart_send_static_text("\"I\"         Get camera parameter\r\n");
uart_send_static_text("\"J,#,#,#,#\" Set camera parameter mode,width,heigth,zoom(1,4 or 8)\r\n");
uart_send_static_text("\"K\"         Calibrate proximity sensors\r\n");
uart_send_static_text("\"L,#,#\"     Led number,0=off 1=on 2=inverse\r\n");
#ifdef FLOOR_SENSORS
```

```
uart_send_static_text("\"M\"         Floor sensors\r\n");
#endif
uart_send_static_text("\"N\"         Proximity\r\n");
uart_send_static_text("\"O\"         Light sensors\r\n");
uart_send_static_text("\"P,#,#\"     Set motor position left,right\r\n");
uart_send_static_text("\"Q\"         Get motor position left,right\r\n");
uart_send_static_text("\"R\"         Reset e-puck\r\n");
uart_send_static_text("\"S\"         Stop e-puck and turn off leds\r\n");
uart_send_static_text("\"T,#\"       Play sound 1-5 else stop sound\r\n");
uart_send_static_text("\"U\"         Get microphone amplitude\r\n");
uart_send_static_text("\"V\"         Version of SerCom\r\n");
#ifdef LIS_SENSORS_TURRET
if (sensext_pres) // If the TP_sensors tourret is present
uart_send_static_text("\"W,#\"       Sensor extension turret. Analog: W,0; analog 5 LEDs: W,0->31; i2c dist: W,-1; i2c comp: W,-2\r\n");
else
uart_send_static_text("\"W,#\"       Sensor extension turret not detected. Function deactivated.");
#endif

break;
case 'I':
sprintf(buffer,"i,%d,%d,%d,%d,%d\r\n",cam_mode,cam_width,cam_heigth,cam_zoom,cam_size);
uart_send_text(buffer);
break;
case 'J'://set camera parameter see also cam library
sscanf(buffer,"J,%d,%d,%d,%d\r",&cam_mode,&cam_width,&cam_heigth,&cam_zoom);
if(cam_mode==GREY_SCALE_MODE)
cam_size=cam_width*cam_heigth;
else
cam_size=cam_width*cam_heigth*2;
e_po3030k_init_cam();
e_po3030k_config_cam((ARRAY_WIDTH -cam_width*cam_zoom)/2,(ARRAY_HEIGHT-cam_heigth*cam_zoom)/2,cam_width*cam_zoom,cam_heigth*cam_zoom,cam_zoom,cam_zoom,cam_mode);
     e_po3030k_set_mirror(1,1);
    e_po3030k_write_cam_registers();
    uart_send_static_text("j\r\n");
    break;
case 'K':  // calibrate proximity sensors
uart_send_static_text("k, Starting calibration - Remove any object in sensors range\r\n");
e_calibrate_ir();
uart_send_static_text("k, Calibration finished\r\n");
break;
case 'L': // set led
sscanf(buffer,"L,%d,%d\r",&LED_nbr,&LED_action);
e_set_led(LED_nbr,LED_action);
uart_send_static_text("l\r\n");
break;
case 'M': // read floor sensors (optional)
#ifdef FLOOR_SENSORS
e_i2cp_enable();
for (i=0; i<6; i++) buffer[i] = e_i2cp_read(0xC0,i);
e_i2cp_disable();
sprintf(buffer,"m,%d,%d,%d\r\n",
(unsigned int)buffer[1] | ((unsigned int)buffer[0] << 8),
(unsigned int)buffer[3] | ((unsigned int)buffer[2] << 8),
(unsigned int)buffer[5] | ((unsigned int)buffer[4] << 8));
uart_send_text(buffer);
#else
uart_send_static_text("m,0,0,0\r\n");
#endif
break;
case 'N': // read proximity sensors
sprintf(buffer,"n,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
        e_get_calibrated_prox(0),e_get_calibrated_prox(1),e_get_calibrated_prox(2),e_get_calibrated_prox(3),
        e_get_calibrated_prox(4),e_get_calibrated_prox(5),e_get_calibrated_prox(6),e_get_calibrated_prox(7));
uart_send_text(buffer);
break;
case 'O': // read ambient light sensors
sprintf(buffer,"o,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
        e_get_ambient_light(0),e_get_ambient_light(1),e_get_ambient_light(2),e_get_ambient_light(3),
        e_get_ambient_light(4),e_get_ambient_light(5),e_get_ambient_light(6),e_get_ambient_light(7));
uart_send_text(buffer);
break;
case 'P': // set motor position
sscanf(buffer,"P,%d,%d\r",&positionl,&positionr);
e_set_steps_left(positionl);
e_set_steps_right(positionr);
uart_send_static_text("p\r\n");
break;
case 'Q': // read motor position
sprintf(buffer,"q,%d,%d\r\n",e_get_steps_left(),e_get_steps_right());
uart_send_text(buffer);
break;
case 'R': // reset
uart_send_static_text("r\r\n");
RESET();
break;
case 'S': // stop
e_set_speed_left(0);
e_set_speed_right(0);
e_set_led(8,0);

uart_send_static_text("s\r\n");
break;
case 'T': // stop
sscanf(buffer,"T,%d",&sound);
if(first==0){
e_init_sound();
first=1;
}
switch(sound)
{
case 1: e_play_sound(0,2112);break;
case 2: e_play_sound(2116,1760);break;
case 3: e_play_sound(3878,3412);break;
case 4: e_play_sound(7294,3732);break;
case 5: e_play_sound(11028,8016);break;
default:
e_close_sound();
```

```
first=0;
break;
}
uart_send_static_text("t\r\n");
break;
case 'U':
sprintf(buffer,"u,%d,%d,%d\r\n",e_get_micro_volume(0),e_get_micro_volume(1),e_get_micro_volume(2));
uart_send_text(buffer);
break;
case 'V': // get version information
uart_send_static_text("v,Version 1.1.4 December 2006\r\n");
break;
case 'W': // read Devantec ultrasonic range sensor or Sharp Ir sensor (optional)
#ifdef LIS_SENSORS_TURRET
sscanf(buffer,"W,%d\r",&sensext_param[0]);
if (sensext_pres) // If the TP_sensors tourret is present
{
if(e_sensext_process(sensext_param, sensext_value))
{
switch (sensext_param[0])
{
case -1: //i2c SRFxxx
sprintf(buffer,"w,%u,%u\r\n", sensext_value[0],  sensext_value[1]);
break;
case -2: // i2c cmps03
sprintf(buffer,"w,%d,%d\r\n",  sensext_value[0],  sensext_value[1]);
break;
default: //analog (sharp,...)
sprintf(buffer,"w,%d\r\n",  sensext_value[0]);

}
uart_send_text(buffer);
}
else
{
uart_send_static_text("wrong parameter\r\n");
}

}
else
{
uart_send_static_text("LIS sensors turret not present\r\n");
}
#endif
break;

case 'X':  // Dummy command returning a number of bytes given as parameter
sscanf(buffer,"X,%d\r",&position1);
buffer[position1+2]='\r';
buffer[position1+3]='\n';
while(position1>0) {
buffer[position1+1]='1';
position1--;
}
buffer[0]='x';
buffer[1]=',';
uart_send_text(buffer);
break;
default:
uart_send_static_text("z,Command not found\r\n");
break;
}
}
}
}
```

1.3 Bluetooth Mirror: program/bluetooth_mirror/main.c

```
#include "./motor_led/e_epuck_ports.h"
#include "stdio.h"
#include "./uart/e_uart_char.h"
#include "./bluetooth/e_bluetooth.h"
#include "./motor_led/e_init_port.h"
#include "string.h"
#include "runbreitenberg.h"

#define uart2_send_static_text(msg) { e_send_uart2_char(msg,sizeof(msg)-1); while(e_uart2_sending()); }

/*this programm is an exemple how to used bluetooth library, if selectore is on 0,
* you can comunicate through normal RS232 conncection (need cable) and test bluetooth function
* other wise e-puck will connect to nearest robot (that must have sercom program) and make a obstacle
* avoidance sending also motor command to the other e-puck*/



int getselector() {
return SELECTOR0 + 2*SELECTOR1 + 4*SELECTOR2 + 8*SELECTOR3;
}


int main(void)
{
 char message[50];


 char command[30], response[80];
 int c;
 int i, version,j;
 int nb_bt_device,error;//max 10


 e_init_port();   //Configure port pins
 e_init_uart1();   //Initialize UART to 115200Kbaud
```

```
 e_init_uart2(); //Initialize UART to 115200Kbaud

 if(RCONbits.POR) //Reset if Power on (some problem for few robots)
 {
RCONbits.POR=0;
__asm__ volatile ("reset");
 }

 if(getselector()==0)//getselector()==0
 {


 sprintf(message, "\n\n\rWELCOME to the protocol on e-Puck Bluetooth calibration");
 e_send_uart2_char(message,strlen(message));  //send string
 sprintf(response, "\n\rPress H (return) for help");
 e_send_uart2_char(response,strlen(response));  //send string


while(1)
{
   i = 0;
c=0;
   do
   {
      if (e_getchar_uart2(&command[i]))
{
c=command[i];
     i++;
}
   }
   while (((char)c != '\n')&&((char)c != '\x0d'));
   command[i]='\0';

   switch (command[0])
   {
    case 'A': sprintf(message,"\n\rSearch begin");
e_send_uart2_char(message,strlen(message));
nb_bt_device=e_bt_find_epuck();
for(j=0;j<nb_bt_device;j++){
sprintf(response,"\n\r%d. e_puck_%c%c%c%c BT_address:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x",j,(unsigned char)e_bt_present_epuck[j].number[0],(unsigned char)e_bt_present_epuck[j].numbe
e_send_uart2_char(response,strlen(response));
while(e_uart2_sending());
}
sprintf(response,"\n\rInquiry finished");
break;
case 'B': sprintf(message,"\n\rSearch e-puck");
e_send_uart2_char(message,strlen(message));
if(e_bt_connect_epuck()==0)
sprintf(response,"\n\rc, connected");
else
sprintf(response,"\n\rc, connection failed");
break;
case 'P': e_bt_read_local_pin_number(message);
sprintf(response,"\n\rPIN code = %s",message);
break;
case 'O': sscanf(command,"O,%s\n",message);
if(e_bt_write_local_pin_number(message))
sprintf(response,"\n\rError writting PIN");
else
sprintf(response,"\n\rPIN code = %s",message);
break;
case 'M': sscanf(command,"M,%s\n",message);
if(e_bt_write_local_name(message))
sprintf(response,"\n\rError writting Name");
else
sprintf(response,"\n\rFriendly name = %s",message);
break;
case 'S': sscanf(command,"S,%s\n",message);
if(e_bt_write_local_pin_number(message))
sprintf(response,"\n\rError writting PIN");
else
sprintf(response,"\n\rPIN code = %s",message);
e_send_uart2_char(response,strlen(response));
sprintf(command,"e-puck_%s",message);
if(e_bt_write_local_name(command))
sprintf(response,"\n\rError writting Name");
else
sprintf(response,"\n\rFriendly name = %s",command);
break;
case 'N': e_bt_read_local_name(message);
sprintf(response,"\n\rFriendly name = %s",message);
break;
case 'R': version=e_bt_reset();
sprintf(response,"\n\rReset ok Firmware = %d",version);
break;
case 'H': uart2_send_static_text("\n\r \"A\" find e-puck");
uart2_send_static_text("\n\r \"B\" connect to nearest e-puck");
uart2_send_static_text("\n\r \"C,#\" Connect SPP device # (make inquiry first)");
uart2_send_static_text("\n\r \"D,\%s\" Send data");
uart2_send_static_text("\n\r \"E\" Release SPP conection");
uart2_send_static_text("\n\r \"F,#\" Ask friendly name local device # (make inquiry first)");
uart2_send_static_text("\n\r \"I\" Inquiry local device");
uart2_send_static_text("\n\r \"K\" List local paired device");
uart2_send_static_text("\n\r \"L,#\" Remove pairing device device # (make list local paired device first)");
uart2_send_static_text("\n\r \"M,Name\" Write Name for Friendly Bluetooth name");
uart2_send_static_text("\n\r \"N\" Read actual Friendly Bluetooth name");
uart2_send_static_text("\n\r \"O,#\" Write # PIN number");
uart2_send_static_text("\n\r \"P\" Read actual PIN number");
uart2_send_static_text("\n\r \"R\" Soft reset Bluetooth module");
uart2_send_static_text("\n\r \"S,#\" Write # PIN number and same time e-puck_#");
uart2_send_static_text("\n\r \"T\" Enter transparent mode");
uart2_send_static_text("\n\r \"U\" Exit transparent mode");

response[0]='\n';
response[1]='\0';
break;
case 'I': sprintf(message,"\n\rInquiry begin");
```

```
e_send_uart2_char(message,strlen(message));
nb_bt_device=e_bt_inquiry(e_bt_present_device);
for(j=0;j<nb_bt_device;j++){
sprintf(response,"\n\rDevice %d BT_address:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x Type:%2.2x:%2.2x:%2.2x",j,(unsigned char)e_bt_present_device[j].address[0],(unsigned char)e_bt_present
e_send_uart2_char(response,strlen(response));
while(e_uart2_sending());
}
sprintf(response,"\n\rInquiry finished");
break;
case 'F': sscanf(command,"F,%d\r",&j);
e_bt_get_friendly_name(&e_bt_present_device[j]);
sprintf(response,"\n\rBT_address:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x Friendly Name:%s",(unsigned char)e_bt_present_device[j].address[0],(unsigned char)e_bt_present_device[j].address
break;
case 'K': nb_bt_device=e_bt_list_local_paired_device();
for(j=0;j<nb_bt_device;j++){
sprintf(response,"\n\rDevice %d BT_address:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x",j,(unsigned char)e_bt_local_paired_device[0+j*6],(unsigned char)e_bt_local_paired_device[1+j*6],( uns
e_send_uart2_char(response,strlen(response));
while(e_uart2_sending());
}
response[0]='\n';
response[1]='\0';
break;
case 'L': sscanf(command,"L,%d\r",&j);
e_bt_remove_local_paired_device(j);
sprintf(response,"\n\r1,erase paring");
break;
case 'D': sscanf(command,"D,%s\r",&message);
e_bt_send_SPP_data(message,strlen(message));
sprintf(response,"\n\rd,data send");
break;
case 'C': sscanf(command,"C,%d\r",&j);
if(e_bt_etablish_SPP_link(&e_bt_present_device[j].address[0])==0)
sprintf(response,"\n\rc, connected");
else
sprintf(response,"\n\rc, connection failed");
break;
case 'E': if(e_bt_release_SPP_link()==0x1f)
sprintf(response,"\n\ry, no connection");
else
sprintf(response,"\n\ry, released");
break;
case 'T': e_bt_tranparent_mode();
sprintf(response,"\n\rt, tranparent mode");
break;
case 'U': e_bt_exit_tranparent_mode();
sprintf(response,"\n\ru, exit tranparent mode");
break;
case 'X': e_bt_factory_reset();
sprintf(response,"\n\rx, factory reset");
break;
    default:    sprintf(response,"\n\rz,Command not found");
                break;
   }
   e_send_uart2_char(response,strlen(response));
   while(e_uart2_sending());
 }
}
else
{
LED0=1;
error=e_bt_connect_epuck();
if(error==0){
e_bt_tranparent_mode();
run_breitenberg();
}
LED3=1;
}
LED0=0;

while(1)
{
LED4!=LED4;
for(j=0;j<30000;j++);
}

return 1;
}
```

1.4 Pre-Installed Demo Programs: program/demo_swis_1

1.4.1 main_c.c

```
/***************************************************************************

Programm to demonstrate all the e-puck's capabilities
Version 2.0 aot 2007
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
```

```
 * \brief The main file of the demo.
 *
 * This file regroups all the programmes demo. You can choose the one you
 * want to try by moving the selector.
 * \n \n Here is the the demo corresponding to the selector position:
 * - Selector position 0: Show the ground direction. Look at rungrounddirection.h for more information.
 * - Selector position 1: Shock detection. Look at runaccelerometer.h for more information.
 * - Selector position 2: Locate the sound source. Look at runlocatesound.h for more information.
 * - Selector position 3: Follow the wall. Look at runwallfollow.h for more information.
 * - Selector position 4: Follow what is detected by the two front proximities detectors. Look at runbreitenberg_adv.h for more information.
 * - Selector position 5: Avoid the obstacles. Look at runbreitenberg_adv.h for more information.
 * - Selector position 6: Follow all balls. Look at runfollowball.h for more information.
 * - Selector position 7: Follow the red ball. Look at runfollowball.h for more information.
 * - Selector position 8: Follow the green ball. Look at runfollowball.h for more information.
 * - Other selector position: Command the e-puck with the sound. Look at runfftlistener.h for more information.
 *
 * \warning When you have made your selection with the selector, YOU MUST reset
 * the e-puck (the blue button near the selector) to make your choice effective.
 * \section sect_selector_pos The selector position:
 * \image html selector.gif
 * \author Code: Michael Bonani, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

/*! \mainpage Demo Swiss 1 documentation
 * \image html logo.gif
 * \section intro_sec Introduction
 * The programm "DemoSwis1" is made to demonstrate some capabilities of
 * the e-puck and to illustrate how you can use the library.
 * \n \n This demo is, in fact, a compilation of nine demos. You can choose
 * the demo you want to play by moving the selector on the back of the e-puck.
 * \n \n Here is the the demo corresponding to the selector position:
 * - Selector position 0: Shock accelerometer. Look at runaccelerometer.h for more information.
 * - Selector position 1: Locate the sound source. Look at runlocatesound.h for more information.
 * - Selector position 2: Follow the wall. Look at runwallfollow.h for more information.
 * - Selector position 3: Show the ground direction. Look at rungrounddirection.h for more information.
 * - Selector position 4: Follow what is detected by the two front proximities detectors. Look at runbreitenberg_adv.h for more information.
 * - Selector position 5: Avoid the obstacles. Look at runbreitenberg_adv.h for more information.
 * - Selector position 6: Follow the green ball. Look at runfollowball.h for more information.
 * - Selector position 7: Follow the red ball. Look at runfollowball.h for more information.
 * - Selector position 8: Follow all balls. Look at runfollowball.h for more information.
 * - Other selector position: Command the e-puck with the sound. Look at runfftlistener.h for more information.
 *
 * \warning When you have made your selection with the selector, YOU MUST reset
 * the e-puck (the blue button near the selector) to make your choice effective.
 * \section sect_selector_pos2 The selector position:
 * \image html selector.gif
 *
 * \section link_sec External links
 * - http://www.e-puck.org/                The official site of the e-puck
 * - https://gna.org/projects/e-puck/      The developpers 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
 *
 * \author Doc: Jonathan Besuchet
 */

#include "p30f6014A.h"
#include "stdio.h"
#include "string.h"

#include "uart/e_uart_char.h"

#include "ComModule.h"
#include "runcollaboration.h"
#include "runaccelerometer.h"
#include "runbreitenberg_adv.h"
#include "runlocatesound.h"
#include "runwallfollow.h"
#include "runfollowball.h"
#include "runfftlistener.h"
#include "utility.h"

#define PI 3.14159265358979

/*! \brief The main function of the application
 *
 * The main function launch one of the following self-governing function
 * depending of the selector position: run_accelerometer(); run_locatesound();
 * run_wallfollow(); run_grounddirection(); run_breitenberg_follower();
 * run_breitenberg_shocker(); run_follow_ball_green(); run_follow_ball_red();
 * run_follow_ball(); run_fft_listener();
 */
int main() {
char buffer[80];
int selector;

//system initialization
e_init_port();
e_init_uart1();

//Reset if Power on (some problem for few robots)
if (RCONbits.POR) {
RCONbits.POR=0;
__asm__ volatile ("reset");
}

// Decide upon program
selector=getselector();
sprintf(buffer, "Starting with selector pos %d\r\n", selector);
e_send_uart1_char(buffer, strlen(buffer));

if (selector==0) {
run_grounddirection();
} else if (selector==1) {
run_accelerometer();
} else if (selector==2) {
run_locatesound();
```

```
} else if (selector==3) {
run_wallfollow();
} else if (selector==4) {
run_breitenberg_follower();
} else if (selector==5) {
run_breitenberg_shocker();
} else if (selector==6) {
run_follow_ball();
} else if (selector==7) {
run_follow_ball_red();
} else if (selector==8) {
run_follow_ball_green();
} else
run_fft_listener();

while(1);
return 0;
}
```

1.4.2 common.inc

```
;START_HEADER
;
; dsPIC30F6014 Demo Source File
; (c) Copyright 2003 Microchip Technology, All rights reserved
;
; ----------------------------------------------------------------------------
; File Revision History:
; ----------------------------------------------------------------------------
;
; $Log: common.inc,v $
; Revision 1.1.1.1  2003/08/23 00:38:33  VasukiH
; First import of demo source into CVS Repository
;
;
;
; ----------------------------------------------------------------------------
;
; Software and Development Tools Info:
; ----------------------------------------------------------------------------
; Tool                          Version
; ----------------------------------------------------------------------------
; MPLAB IDE                     6.30
; MPLAB C30 Toolsuite           1.10.02
; dsPICDEM QFP Processor Board  1.10
; ----------------------------------------------------------------------------
;
; File Notes:
;
;
;END_HEADER


; Data Size Unit constants
.equ    WORD,           2
.equ    CPLXWORD,       4
.equ    BYTE,           1

;Constants used to initialize the DCI module
.equ  Fs, 8000
.equ  FSCKD, Fs * 256                         ; frame clock rate
.equ  Fcy, 7372800*2                          ; device instruction rate
.equ  BCG,  ( Fcy / ( 2 * FSCKD ) ) - 1       ; equation for DCI clock rate

;Constants used while playing DTMF tones via the DCI module
;.equ NUMSAMPSTORED, 15704
.equ NUMSAMPSTORED, 13204

;Constants useful to initialize Si3000 Codec
.equ  minus_one, 0x8000
.equ  minus_one_with_secondary,  0x8001

.equ  plus_one, 0x7FFE
.equ  plus_one_with_secondary,   0x7FFF

; Si3000 Register Address Summary (Table 13, Si30000-DS11)
; Read Control Address has bit 13 set
; Write Control Address has bit 13 clear

.equ  Read_Control_1,           0x2100
.equ  Write_Control_1,          0x0100

.equ  Read_Control_2,           0x2200
.equ  Write_Control_2,          0x0200

.equ  Read_PLL1_Divide_N1,      0x2300
.equ  Write_PLL1_Divide_N1,     0x0300

.equ  Read_PLL1_Multiply_M1,    0x2400
.equ  Write_PLL1_Multiply_M1,   0x0400

.equ  Read_RX_Gain_Control_1,   0x2500
.equ  Write_RX_Gain_Control_1,  0x0500

.equ  Read_ADC_Volume_Control,  0x2600
.equ  Write_ADC_Volume_Control, 0x0600

.equ  Read_DAC_Volume_Control,  0x2700
.equ  Write_DAC_Volume_Control, 0x0700

.equ  Read_Status_Report,       0x2800
.equ  Write_Status_Report,      0x0800

.equ  Read_Analog_Attenuation,  0x2900
```

```
.equ  Write_Analog_Attenuation, 0x0900
```

1.4.3 ComModule.{h,c}

```
#ifndef _COM_MODULE
#define _COM_MODULE


#define COM_MODULE_HW_ATTENUATOR_25DB 1
#define COM_MODULE_HW_ATTENUATOR_0DB 0
#define COM_MODULE_DEFAULT_GROUP 0x7d

#define COM_MODULE_MAXSIZE  28

#include "p30f6014A.h"

void InitComModule(unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue);
// OwnGroup : module will only receive packet of that group ID.
// hardwareAttenuator : COM_MODULE_HW_ATTENUATOR_25DB or COM_MODULE_HW_ATTENUATOR_0DB
// softAttenuator : value from 1 (full power) to 31 (-25dB)

int IsModulePlugged(); // return 0 if nothing connected, 1 if module is here.

void SetRadioEnabledState(unsigned char mode);// enable (1) or disable (0) the radio part of the module
void SetHardwareAttenuator(unsigned char AttenuatorMode); // COM_MODULE_HW_ATTENUATOR_25DB or COM_MODULE_HW_ATTENUATOR_0DB
void SetSoftwareAttenuator(unsigned char AttenuatorValue); // value from 1 (full power) to 31 (-25dB)
void SetOwnGroup(unsigned char GroupID); // module will only receive packet of that group ID.
void SetOwnAddress(unsigned int ownaddress); // current address of the module

unsigned char GetRadioEnabledState(); // renturn if radio in enabled or not;
unsigned char GetHardwareAttenuator(); // return COM_MODULE_HW_ATTENUATOR_25DB or COM_MODULE_HW_ATTENUATOR_0DB
unsigned char GetSoftwareAttenuator(); // return value from 1 (full power) to 31 (-25dB)
unsigned char GetOwnGroup(); // return module will only receive packet of that group ID.

unsigned char GetStatus(); // return status register
//bit 0 : PACKET_READY_FLAG : a packet is in receive buffer. auto clear when last byte of receive buffer is read
//bit 1 : TX_IDLE_FLAG  : module is ready to send a new packet
//bit 2 : PACKET_LOST_FLAG  : a packet was in receive buffer when another packet arrived. 2.nd packet is lost. auto clear when last byte of receive buffer is read
//bit 3 : TX_SEND_ERROR : error durint packet send. auto clear during next send attempt.

int SendPacket(unsigned char destinationgroup, unsigned int destinationaddress, unsigned char* packet, int packetsize); // return 1 if OK, 0 if error
// destinationGroup : the destination group of the packet 0xFF is broadcast group
// destinationaddress : the address of the destination module. 0xff is broadcast address
// packet : unsigned char array of data to be sent
// packetSize : number of fields of the unsigned char array to be sent. ( max is COM_MODULE_MAXSIZE bytes )

int IsPacketReady(unsigned char* packet, int* packetSize); // return 1 and the packet + size if new packet receive. 0 if not.
// packet : unsigned char array of data to put receive data in. must be COM_MODULE_MAXSIZE bytes array.
// packetSize : the effective data size receive.

#endif
```

```
#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"


#define COM_MODULE_I2C_ADDR (0x3F<<1)

#define STATUS_REG_ADDR 0x00  // status like PACKET_LOST_FLAG,TX_IDLE,PACKET_READY
#define CONFIG_REG_ADDR 0x01  // config like ..... hardware att. sleep...
#define SEND_REG_ADDR 0x02  // <> 0 -> send sendbuffer
#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 0x31
#define REC_BUFFER_START 0x32
#define REC_BUFFER_END 0x5C

#define AM_MSGTYPE 0x0A // AM_OSCOPE

#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 // how many byte after packet start should I write data ?

//status reg
#define PACKET_READY_FLAG 0x01 // bit 1
#define TX_IDLE_FLAG 0x02 // bit 2
#define PACKET_LOST_FLAG 0x04 // bit 3
#define TX_SEND_ERROR 0x08 // bit 4

//config reg
#define HARDWAREATT_SET_FLAG 0x01 //bit 1
#define RADIO_ENABLED_FLAG 0x80 //bit 8

// send reg
#define REQUEST_TO_SEND_FLAG 0x01

unsigned char ReadRegister(unsigned char registeraddr) {
return (0x00FF & e_i2cp_read(COM_MODULE_I2C_ADDR, registeraddr));
}

void WriteRegister(unsigned char registeraddr, unsigned char value) {
```

```
e_i2cp_write(COM_MODULE_I2C_ADDR, registeraddr, value);
}

void InitComModule(unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue) {
e_i2cp_init();
e_i2cp_enable();

while (IsModulePlugged() == 0); // wait till module online

SetOwnGroup(owngroup);
SetOwnAddress(ownaddress);
SetHardwareAttenuator(hardwareattenuatormode);
SetSoftwareAttenuator(softwareattenuatorvalue);
SetRadioEnabledState(1);
}

int IsModulePlugged() {
unsigned char value;
value = GetStatus();
if (value == 0xFF) // value read are always 0xff when module not plugged in
return 0;
return 1;
}

void SetRadioEnabledState(unsigned char mode) {
unsigned char old_value;
old_value = ReadRegister(CONFIG_REG_ADDR);
if (mode != 0)
WriteRegister(CONFIG_REG_ADDR, old_value |= RADIO_ENABLED_FLAG);
else
WriteRegister(CONFIG_REG_ADDR, old_value &= ~RADIO_ENABLED_FLAG);
}

void SetHardwareAttenuator(unsigned char attenuatormode) {
unsigned char old_value;
old_value = ReadRegister(CONFIG_REG_ADDR);
if (attenuatormode != 0)
WriteRegister(CONFIG_REG_ADDR, old_value |= HARDWAREATT_SET_FLAG);
else
WriteRegister(CONFIG_REG_ADDR, old_value &= ~HARDWAREATT_SET_FLAG);
}


void SetSoftwareAttenuator(unsigned char attenuatorvalue) {
if (attenuatorvalue > 31)
attenuatorvalue = 31;
WriteRegister(SOFTATT_REG_ADDR, attenuatorvalue);
}

void SetOwnGroup(unsigned char owngroup) {
WriteRegister(OWNGROUP_REG_ADDR, owngroup);
}

void SetOwnAddress(unsigned int ownaddress) {
WriteRegister(OWNADDRL_REG_ADDR, ownaddress & 0xFF);
ownaddress=ownaddress>>8;
WriteRegister(OWNADDRH_REG_ADDR, ownaddress & 0xFF);
}

unsigned char GetHardwareAttenuator() {
if (ReadRegister(CONFIG_REG_ADDR) & HARDWAREATT_SET_FLAG);
return 1;
return 0;
}

unsigned char GetRadioEnabledState() {
if (ReadRegister(CONFIG_REG_ADDR) & RADIO_ENABLED_FLAG);
return 1;
return 0;
}

unsigned char GetSoftwareAttenuator() {
return ReadRegister(SOFTATT_REG_ADDR);
}

unsigned char GetOwnGroup() {
return ReadRegister(OWNGROUP_REG_ADDR);
}

unsigned int GetOwnAddress() {
unsigned char lowbyte, highbyte;
lowbyte=ReadRegister(OWNADDRL_REG_ADDR);
highbyte=ReadRegister(OWNADDRH_REG_ADDR);
return lowbyte | ((int)highbyte<<8);
}

unsigned char GetStatus() {
return ReadRegister(STATUS_REG_ADDR);
}

int SendPacket(unsigned char destinationgroup, unsigned int destinationaddress, unsigned char* packet, int packetsize) {
unsigned int i;
if (packetsize > COM_MODULE_MAXSIZE)
return 0; // not allowed -> return error

while ((GetStatus() & TX_IDLE_FLAG) == 0); // wait till module is ready to transmit

WriteRegister(SEND_BUFFER_START + AM_MSGTYPE_IN_PACKET_OFFSET, AM_MSGTYPE);
WriteRegister(SEND_BUFFER_START + GROUPDATA_IN_PACKET_OFFSET, destinationgroup);
WriteRegister(SEND_BUFFER_START + ADDRLDATA_IN_PACKET_OFFSET, destinationaddress & 0xFF);
destinationaddress=destinationaddress>>8;
WriteRegister(SEND_BUFFER_START + ADDRHDATA_IN_PACKET_OFFSET, destinationaddress & 0xFF);
for (i=0; i<packetsize; i++) {
WriteRegister(FIRSTDATA_IN_PACKET_OFFSET+SEND_BUFFER_START+i, packet[i]);
}
WriteRegister(SEND_REG_ADDR,1); // send packet
// maybe wait here
```

```
if (GetStatus() & TX_SEND_ERROR) // flag TX error set -> return error
return 0;
return 1; // return OK
}

int IsPacketReady(unsigned char* packet, int* packetsize) {
unsigned int i;
if ((packet == NULL) || (packetsize == NULL))
return 0; // return withour reading NULL pointer means uC reset if write
if (GetStatus() & PACKET_READY_FLAG) { // a packet is ready !
*packetsize = ReadRegister(REC_BUFFER_START); // get size of received buffer
if (*packetsize > COM_MODULE_MAXSIZE)
return 0; // error in transmission. size is too big and impossible
for (i = 0; i< *packetsize ; i++) {
packet[i] = ReadRegister(REC_BUFFER_START+FIRSTDATA_IN_PACKET_OFFSET+i);
}
ReadRegister(REC_BUFFER_END);
return 1;
}
for(i=0; i<0x1000; i++); // wait
return 0; // no packet ready -> 0
}
```

1.4.4 runaccelerometer.{h,c}

```
/*****************************************************************************
*  Programm which detects the shocks enduring by the e-puck      *
* Version 2.0 aot 2007                                *
* Michael Bonani, Jonathan Besuchet                *
*                                    *
*****************************************************************************/

/*! \file
 * \brief The shocks detector
 * \section sect1 Introduction
 * This demo uses the acceleromter to detect the direction of the shocks
 * enduring by the e-puck.
 * \n When the e-puck has deteced a shock, he plays a sound and show you
 * the direction of this shock by turning on the appropriated LED.
 *
 * \section sect2 Playing the demo
 * First of all, move the selector to the position 0 and reset the e-puck.
 * \n Then you only have to hit the e-puck and a LED will turn on to
 * show you the direction of the shock that the e-puck has detected. The
 * e-puck will also play a sound.
 * \n If you let the e-puck falling, he detects the "zero_G" and beginn to
 * cry.
 *
 * \section sect3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=NCK9gRL9mb4
 *
 * \author Jonathan Besuchet
 */

#ifndef _ACCELEROMETER_SHOKCS
#define _ACCELEROMETER_SHOCKS

int run_accelerometer();

#endif
```

```
/*****************************************************************************

Programm which detects the shocks enduring by the e-puck
Version 2.0 aot 2007
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \brief The shocks detector
 * \section sect1 Introduction
 * This demo uses the acceleromter to detect the direction of the shocks
 * enduring by the e-puck.
 * \n When the e-puck has deteced a shock, he plays a sound and show you
 * the direction of this shock by turning on the appropriated LED.
 *
 * \section sect2 Playing the demo
 * First of all, move the selector to the position 0 and reset the e-puck.
 * \n Then you only have to hit the e-puck and a LED will turn on to
 * show you the direction of the shock that the e-puck has detected. The
 * e-puck will also play a sound.
 * \n If you let the e-puck falling, he detects the "zero_G" and beginn to
 * cry.
 *
 * \section sect3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=NCK9gRL9mb4
 *
 * \author Code: Michael Bonani, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */
```

```
#include "p30f6014A.h"
#include "stdio.h"
#include "string.h"

#include "codec/e_sound.h"
#include "motor_led/e_init_port.h"
#include "motor_led/advance_one_timer/e_led.h"
#include "motor_led/advance_one_timer/e_motors.h"
#include "uart/e_uart_char.h"
#include "a_d/advance_ad_scan/e_ad_conv.h"
#include "a_d/advance_ad_scan/e_acc.h"

#include "math.h"
#include "utility.h"

#define STATE_NORMAL (0)
#define STATE_FREEFALL (1)
#define STATE_SHOCK (2)

#define PI 3.14159265358979

//#define MOVE //option if you want that e-puck move

extern int e_dci_unavailable;
extern int e_stop_flag;


/*! \brief The "main" function of the demo */
void run_accelerometer() {
int accx, accy, accz;
long ampl;
long amplavg;
char buffer[80];
int state;
int lednum;
double angle;
int soundsel;

// Init sound
e_init_port();
    e_init_ad_scan(ALL_ADC);
e_init_sound();
#ifdef MOVE
e_init_motors();
e_start_agendas_processing();
#endif

// Calibrate accelerometers
e_set_led(8, 1);
e_acc_calibr();
e_set_led(8, 0);

#ifdef MOVE
// Move forwards
e_set_speed_left(100);
e_set_speed_right(100);
#endif
// Detect free fall and shocks
state=STATE_NORMAL;
amplavg=1000;
while (1) {

accx = e_get_acc(0);
accy = e_get_acc(1);
accz = e_get_acc(2) + 744; //744 is lg

if ((accz<0) && (accz>-600)) {accz=0;}
ampl=((long)(accx)*(long)(accx))+((long)(accy)*(long)(accy))+((long)(accz)*(long)(accz));
amplavg=(amplavg>>2)+ampl;

if (! e_dci_unavailable) {
if (state!=STATE_NORMAL) {
state=STATE_NORMAL;
e_set_led(8, 0);
e_set_body_led(0);
}
}

if (amplavg<1000) {
if (state!=STATE_FREEFALL) {
state=STATE_FREEFALL;
e_stop_flag=1;
while (e_dci_unavailable);
sprintf(buffer, "Free fall: %ld, (%d, %d, %d) -> (%ld)\r\n", amplavg, accx, accy, accz, ampl);
e_send_uart1_char(buffer, strlen(buffer));
e_play_sound(11028, 8016);
e_set_body_led(1);
e_set_led(8, 0);
}
} else if (amplavg>4000000) {
if (state!=STATE_SHOCK) {
state=STATE_SHOCK;
e_stop_flag=1;
while (e_dci_unavailable);
sprintf(buffer, "Shock: %ld, (%d, %d, %d) -> (%ld)\r\n", amplavg, accx, accy, accz, ampl);
e_send_uart1_char(buffer, strlen(buffer));
soundsel=(accx & 3);
if (soundsel==0) {
e_play_sound(0, 2112);
} else if (soundsel==1) {
e_play_sound(2116, 1760);
} else {
e_play_sound(3878, 3412);
}
e_set_body_led(0);
```

```
angle=atan2(accy, accx);
lednum=floor(atan2(accy, accx)/PI*4+PI/2+PI/8);
while (lednum>8) {lednum=lednum-8;}
while (lednum<0) {lednum=lednum+8;}
sprintf(buffer, "(x=%d, y=%d) -> angle=%f, led=%d\r\n", accx, accy, angle, lednum);
e_send_uart1_char(buffer, strlen(buffer));
e_set_led(lednum, 1);
}
}
}
}
```

1.4.5 runacclive.{h,c}

```
int run_acclive();
```

```
#include "p30f6014A.h"
#include "stdio.h"
#include "string.h"

#include "../../../library/motor_led/e_init_port.h"
#include "../../../library/a_d/e_accelerometer.h"
#include "../../../library/motor_led/e_led.h"
#include "../../../library/uart/e_uart_char.h"
#include "../../../library/a_d/e_ad_conv.h"
#include "../../../library/a_d/e_prox.h"
#include "utility.h"

void run_acclive() {
int i;
int accx, accy, accz;
int saccx, saccy, saccz;
char buffer[80];

e_init_acc();

while (1) {
saccx=0;
saccy=0;
saccz=0;
for(i=0;i<8;i++) {
e_get_acc(&accx, &accy, &accz);
saccx+=accx;
saccy+=accy;
saccz+=accz;
wait(500);
}

sprintf(buffer, "%d, %d, %d\r\n", saccx, saccy, saccz);
e_send_uart1_char(buffer, strlen(buffer));
wait(10000);
}
}
```

1.4.6 runbreitenberg_adv.{h,c}

```
/*****************************************************************************
*  Programm to follow/avoid obstacles           *
* Version 2.0 aot 2007                           *
* Michael Bonani, Jonathan Besuchet               *
*                                  *
*****************************************************************************/

/*! \file
 * \brief Obstacles follower/avoider
 * \section sect1 Introduction
 * This demo is made to illustrate how to use the 8 proximities sensors.
 * \n The demo is divided in two programms:
 * - A programm to follow what is detected on the front of the e-puck
 * (for exemple another e-puck).
 * - A programm to avoid the obstacles.
 *
 * These two programms work exactelly on the same way. First we do the
 * acquisition of the proxymities sensor and then we do a level-headed
 * sum of this acquisition. Finally we set the speed in concordance of
 * the level-headed sum.
 *
 * \section sect2 Playing the demo
 * \subsection subsect1 Playing the follower
 * First of all, move the selector to the position 4 and reset the e-puck.
 * The e-puck will go forward until he meets anything (it works good with
 * fingers or another e-puck). After that he will
 * follow him by staying always on the same distance.
 * \subsection subsect2 Playing the avoider
 * First of all, move the selector to the position 5 and reset the e-puck.
 * The e-puck will go forward until he meets an obstacle. At this time he
 * will find the better way to avoid him.
 *
 * \section sect3 Video of the demo
 * - The video of the follower demo: http://www.youtube.com/watch?v=6DYp5Olcnew
 * - The video of the avoider demo: http://www.youtube.com/watch?v=Y-RsvDyUfUE
 *
 * \author Jonathan Besuchet
 */

#ifndef _BREITENBERG
```

```
#define _BREITENBERG

void run_breitenberg_follower(void);
void run_breitenberg_shocker(void);

#endif
```

```
/*****************************************************************************

Programm to follow/avoid obstacles
Version 2.0 aot 2007
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \brief Obstacles follower/avoider
 * \section sect1 Introduction
 * This demo is made to illustrate how to use the 8 proximities sensors.
 * \n The demo is divided in two programms:
 * - A programm to follow what is detected on the front of the e-puck
 * (for exemple another e-puck).
 * - A programm to avoid the obstacles.
 *
 * These two programms work exactelly on the same way. First we do the
 * acquisition of the proxymities sensor and then we do a level-headed
 * sum of this acquisition. Finally we set the speed in concordance of
 * the level-headed sum.
 *
 * \section sect2 Playing the demo
 * \subsection subsect1 Playing the follower
 * First of all, move the selector to the position 4 and reset the e-puck.
 * The e-puck will go forward until he meets anything (it works good with
 * fingers or another e-puck). After that he will
 * follow him by staying always on the same distance.
 * \subsection subsect2 Playing the avoider
 * First of all, move the selector to the position 5 and reset the e-puck.
 * The e-puck will go forward until he meets an obstacle. At this time he
 * will find the better way to avoid him.
 *
 * \section sect3 Video of the demo
 * - The video of the follower demo: http://www.youtube.com/watch?v=6DYp5Olcnew
 * - The video of the avoider demo: http://www.youtube.com/watch?v=Y-RsvDyUfUE
 *
 * \author Code: Michael Bonani, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

#include "motor_led/e_init_port.h"
#include "motor_led/advance_one_timer/e_motors.h"
#include "motor_led/advance_one_timer/e_led.h"
#include "motor_led/advance_one_timer/e_agenda.h"
#include "uart/e_uart_char.h"
#include "a_d/advance_ad_scan/e_ad_conv.h"
#include "a_d/advance_ad_scan/e_prox.h"

#include "./runbreitenberg_adv.h"

#define PROXSCALING_FOLLOW 20
#define PROXSCALING_SHOCK 4
#define BASICSPEED 600

int i, s, m;
long potential[2];
int speed[2];
long ProxSensOffBuf[8];

int factor_array[2][8] =
{{-10,-30,15,0,0,-15,30,10},
{10,30,-15,0,0,15,-30,-10}};

int matrix_prox[2][8] =
{{8,4,2,0,0,-4,-8,-16},
{-16,-8,-4,0,0,2,4,8}};

/*! \brief Calcul the speed to set on each wheel for avoiding
 *
 * Here we do a level-headed sum to take advantage of each captor
 * depending of there position. For exemple if the captor number 0
 * detect something, he has to set the right speed high and set
 * the left speed low.
 */
void shock_neuron(void)
{
for (m = 0; m < 2; m++)
{
potential[m] = 0;
for (s = 0; s < 8; s++)
potential[m] += (matrix_prox[m][s]*e_get_calibrated_prox(s)); // get values from proximity sensors
        speed[m] = (potential[m]/PROXSCALING_SHOCK + BASICSPEED);
}

if((speed[1] < 50 && speed[1] > -50)
```

```
&& (speed[0] < 50 && speed[0] > -50)) {
speed[1] = speed[1] * 20;
speed[0] = speed[0] * 20;
}

if (speed[1] > 1000)
speed[1] = 1000;
else if (speed[1] < -1000 )
speed[1] = -1000;

if (speed[0] > 1000)
speed[0] = 1000;
else if (speed[0] < -1000 )
speed[0] = -1000;

e_set_speed_left(speed[1]);
e_set_speed_right(speed[0]);
}

/*! \brief Calcul the speed to set on each wheel for following
 *
 * Here we do a level-headed sum to take advantage of each captor
 * depending of there position. For exemple if the captor number 0
 * detect something, he has to set the left speed high and set
 * the right speed low.
 */
void follow_neuron(void)
{
int basic_speed;

speed[0] = 0;
speed[1] = 0;

for (m = 0; m < 2; m++)
for (i = 0; i<8; i++)
speed[m]  += e_get_calibrated_prox(i)*factor_array[m][i];

basic_speed = 1400 - (e_get_calibrated_prox(7) + e_get_calibrated_prox(0))*5;
//basic_speed = 1600 - (e_get_prox(7)-ProxSensOffBuf[7] + e_get_prox(0)-ProxSensOffBuf[0]);
speed[1] = basic_speed + (speed[1]/PROXSCALING_FOLLOW);
speed[0] = basic_speed + (speed[0]/PROXSCALING_FOLLOW);

if (speed[0] > 1000)
speed[0] = 1000;
else if ( speed[0] < -1000 )
speed[0] = -1000;
if (speed[1] > 1000)
speed[1] = 1000;
else if ( speed[1] < -1000 )
speed[1] = -1000;
e_set_speed_left(speed[1]);
e_set_speed_right(speed[0]);
}

/*! \brief The "main" function of the follower demo */
void run_breitenberg_follower(void)
{
e_init_port();
e_init_motors();
e_init_ad_scan(ALL_ADC);

e_calibrate_ir();

e_activate_agenda(k2000_led, 2500);
e_activate_agenda(follow_neuron, 650);
e_start_agendas_processing();
while(1);
}

/*! \brief The "main" function of the avoider demo */
void run_breitenberg_shocker(void)
{
e_init_port();
e_init_motors();
e_init_ad_scan(ALL_ADC);

e_calibrate_ir();

e_activate_agenda(flow_led, 900);
e_activate_agenda(shock_neuron, 650);
e_start_agendas_processing();
while(1);
}
```

1.4.7 runbreitenberg.{h,c}

```
int run_breitenberg();
```

```
#include "p30f6014A.h"
#include "stdio.h"
#include "string.h"

#include "../../../library/codec/e_sound.h"
#include "../../../library/motor_led/e_init_port.h"
#include "../../../library/motor_led/e_motors.h"
#include "../../../library/motor_led/e_led.h"
#include "../../../library/uart/e_uart_char.h"
#include "../../../library/a_d/e_ad_conv.h"
#include "../../../library/a_d/e_prox.h"
#include "math.h"
```

```
#include "utility.h"

int sensorzero[8];
int weightleft[8] = {-10,-10,-5,0,0,5,10,10};
int weightright[8] = {10,10,5,0,0,-5,-10,-10};

void sensor_calibrate() {
int i, j;
char buffer[80];
long sensor[8];

for (i=0; i<8; i++) {
sensor[i]=0;
}

for (j=0; j<32; j++) {
for (i=0; i<8; i++) {
sensor[i]+=e_get_prox(i);
}
wait(10000);
}

for (i=0; i<8; i++) {
sensorzero[i]=(sensor[i]>>5);
sprintf(buffer, "%d, ", sensorzero[i]);
e_send_uart1_char(buffer, strlen(buffer));
}

sprintf(buffer, " calibration done\r\n");
e_send_uart1_char(buffer, strlen(buffer));
wait(100000);
}

void run_breitenberg() {
int i;
int sensor;
char buffer[80];
int leftwheel, rightwheel;

// Init sensors
e_init_port();
e_init_motors();
e_init_prox();

// Calibrate sensors
e_set_led(8, 1);
sensor_calibrate();
e_set_led(8, 0);

//
while (1) {
leftwheel=200;
rightwheel=200;
for (i=0; i<8; i++) {
sensor=e_get_prox(i)-sensorzero[i];
sprintf(buffer, "%d, ", sensor);
e_send_uart1_char(buffer, strlen(buffer));
leftwheel+=weightleft[i]*(sensor>>4);
rightwheel+=weightright[i]*(sensor>>4);
}

sprintf(buffer, "setspeed %d %d\r\n", leftwheel, rightwheel);
e_send_uart1_char(buffer, strlen(buffer));

if (leftwheel>800) {leftwheel=800;}
if (rightwheel>800) {rightwheel=800;}
if (leftwheel<-800) {leftwheel=-800;}
if (rightwheel<-800) {rightwheel=-800;}
e_set_speed_left(leftwheel);
e_set_speed_right(rightwheel);
wait(100000);
}
}
```

1.4.8 runcollaboration.{h,c}

```
int run_collaboration(void);
```

```
#include "p30f6014A.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "./motor_led/e_epuck_ports.h"
#include "./motor_led/e_init_port.h"
#include "./motor_led/e_led.h"
#include "./a_d/e_prox.h"
#include "./motor_led/e_motors.h"
#include "ComModule.h"
#include "utility.h"
#include "./uart/e_uart_char.h"

#include "runcollaboration.h"

#define FROM_ROBOT_PKT_ID  0  // ID of packet sent from epuck
#define FROM_MOTE_PKT_ID 1  // ID of packet sent from mote to epuck
   // ID of packet between two motes have other IDs

#define LEFT_FOLLOW 0 // behaviors IDs
#define RIGHT_FOLLOW 1
```

```
#define NB_SENSORS          8 // number of sensors
#define BIAS_SPEED       200 // robot bias speed
#define SENSOR_THRESHOLD 300 // discount sensor noise below threshold
#define MAXSPEED  800 // maximum robot speed

#define COM_RANGE 3   // Radio power from 1 to 31; arena range = ~3
#define OPCHANGE_PROB 25 // Probability of changing opinion * 100
#define COM_TIME 1000 // Time between decision and packet send (in miliseconds)

int opinion; // actual behavior : LEFT_FOLLOW for left wall following, RIGHT_FOLLOW for right
int leftPacketCounter; // # packets received of type "turning left"
int rightPacketCounter; // # packets received of type "turning right"

int HaveToSendOpinion; // 1 if opinion changed 0 other, this flag is used to tell the main loop to send data with the radio

//int Interconn[16] = {6,4,6,3,5,-5,-15,-20,   -18,-15,-5,5,3,4,4,6};
//int Interconn[16] = {5,1,2,3,5,4,4,4,-12,    -13,-5,-1,3,-5,-15,-23};

int collab_sensorzero[8];
int collab_weightleft[8] = {-10,-10,-5,0,0,5,10,10};
int collab_weightright[8] = {10,10,5,0,0,-5,-10,-10};

int getSelectorValue();
int sendLabPacket(int dataA, int dataB);

void collab_sensor_calibrate() {
int i, j;
char buffer[80];
long sensor[8];

for (i=0; i<8; i++) {
sensor[i]=0;
}

for (j=0; j<32; j++) {
for (i=0; i<8; i++) {
sensor[i]+=e_get_prox(i);
}
wait(10000);
}

for (i=0; i<8; i++) {
collab_sensorzero[i]=(sensor[i]>>5);
sprintf(buffer, "%d, ", collab_sensorzero[i]);
e_send_uart1_char(buffer, strlen(buffer));
}

sprintf(buffer, " calibration done\r\n");
e_send_uart1_char(buffer, strlen(buffer));
wait(100000);
}


// timer 1 interrupt
// this code is executed every COM_TIME ms
void __attribute__((interrupt, auto_psv, shadow))
_T1Interrupt(void) {
int nmsg; // Number of message received
int OppositeOp; // the opposit opinion
int OppositeNb; // the number of packet of opposit opinion

IFS0bits.T1IF = 0; // clear interrupt flag

e_set_led(0,2);  // front LED blinks when decision is done
e_set_body_led(0);

switch (getSelectorValue()) {
case 13: // the robot is forced to go right
opinion = RIGHT_FOLLOW;
break;
case 11: // the robot is forced to go left
opinion = LEFT_FOLLOW;
break;
case 12: // normal operation
   nmsg = leftPacketCounter + rightPacketCounter;

   if (opinion == LEFT_FOLLOW) {
OppositeOp = RIGHT_FOLLOW;
OppositeNb = rightPacketCounter;
} else {
OppositeOp = LEFT_FOLLOW;
OppositeNb = leftPacketCounter;
}

   // Check if majority of opinions are opposing
   if (OppositeNb > nmsg/2) {
     // Random chance of changing
     if (rand() < OPCHANGE_PROB*(RAND_MAX/100.0)) {
opinion = OppositeOp;
}
   }
break;
default: // wrong position
e_set_body_led(1);
break;
}

HaveToSendOpinion = 1;
leftPacketCounter = 0;
rightPacketCounter = 0;
}


/* init the Timer 1 */
void InitTMR1(void) {
  T1CON = 0;
```

```
  T1CONbits.TCKPS = 3;           // prescsaler = 256
  TMR1 = 0;                      // clear timer 1
  PR1 = (COM_TIME*MILLISEC)/256.0; // interrupt after COM_TIME ms
  IFS0bits.T1IF = 0;             // clear interrupt flag
  IEC0bits.T1IE = 1;             // set interrupt enable bit
  T1CONbits.TON = 1;             // start Timer1
}


/* return the selector value  */
/* from 0 to 15  */
int getSelectorValue() {
return SELECTOR0 + 2*SELECTOR1 + 4*SELECTOR2 + 8*SELECTOR3;
}

/* send two int_16 to default group  */
/* return 1 if OK   */
int sendLabPacket(int dataA, int dataB) {
unsigned char packet[6];
packet[0] = FROM_ROBOT_PKT_ID;
packet[1] = 0;
packet[2] = (unsigned char)((dataA & 0xFF00)>>8);
packet[3] = (unsigned char)(dataA & 0xFF);
packet[4] = (unsigned char)((dataB & 0xFF00)>>8);
packet[5] = (unsigned char)(dataB & 0xFF);
if (SendPacket(COM_MODULE_DEFAULT_GROUP,0xff,packet,6)) {
return 1;
}
return 0;
}

/* return 1 if packet received */
int LabPacketReady(int* dataA, int* dataB) {
unsigned int size;
unsigned char packet[COM_MODULE_MAXSIZE];
if (IsPacketReady(packet, &size) )
{
if ((packet[0] == FROM_ROBOT_PKT_ID) || (packet[0] == FROM_MOTE_PKT_ID))
{
*dataA = (int)(packet[2])<<8 | (int)(packet[3]);
*dataB = (int)(packet[4])<<8 | (int)(packet[5]);
return 1;
}
}
return 0;
}


/*
read sensor prox and return values in a int array
    return values from 0x0000 to 0x00FF (from 0 to 255)
*/
void GetSensorValues(int *sensorTable) {
unsigned int i;
for (i=0; i < NB_SENSORS; i++) {
sensorTable[i] = e_get_prox(i) - collab_sensorzero[i];
}
}

/*
set robot speed
speeds: from -MAXSPEED to MAXSPEED
*/
void setSpeed(int LeftSpeed, int RightSpeed) {
if (LeftSpeed < -MAXSPEED) {LeftSpeed = -MAXSPEED;}
if (LeftSpeed >  MAXSPEED) {LeftSpeed =  MAXSPEED;}
if (RightSpeed < -MAXSPEED) {RightSpeed = -MAXSPEED;}
if (RightSpeed >  MAXSPEED) {RightSpeed =  MAXSPEED;}
e_set_speed_left(LeftSpeed);
e_set_speed_right(RightSpeed);
}

int run_collaboration(void) {
int leftwheel, rightwheel; // motor speed left and right
int distances[NB_SENSORS]; // array keeping the distance sensor readings
int i; // FOR-loop counters
int opinionMessage; // the Opinion receive with the radio
int gostraight;
int loopcount;

leftPacketCounter = 0;
rightPacketCounter = 0;
HaveToSendOpinion = 0;

e_init_port();
e_init_motors();
e_init_prox();
InitComModule(COM_MODULE_DEFAULT_GROUP,10,COM_MODULE_HW_ATTENUATOR_25DB, COM_RANGE );

collab_sensor_calibrate();

// random choice of initial condition (use the LSB of sensor 0 to choose initial direction)
if (e_get_prox(0) & 0x01) {
opinion = LEFT_FOLLOW;
} else {
opinion = RIGHT_FOLLOW;
}

InitTMR1(); // enable inerrupt every COM_TIME ms

loopcount=0;
while (1) {
GetSensorValues(distances); // read sensor values

gostraight=0;
if (opinion == RIGHT_FOLLOW) {
e_set_led(6,0);
```

```
e_set_led(2,1);
for (i=0; i<8; i++) {
if (distances[i]>50) {break;}
}
if (i==8) {
gostraight=1;
} else {
collab_weightleft[0]=-10;
collab_weightleft[7]=-10;
collab_weightright[0]=10;
collab_weightright[7]=10;
if (distances[2]>300) {
distances[1]-=200;
distances[2]-=600;
distances[3]-=100;
}
}
} else {
e_set_led(6,1);
e_set_led(2,0);
for (i=0; i<8; i++) {
if (distances[i]>50) {break;}
}
if (i==8) {
gostraight=1;
} else {
collab_weightleft[0]=10;
collab_weightleft[7]=10;
collab_weightright[0]=-10;
collab_weightright[7]=-10;
if (distances[5]>300) {
distances[4]-=100;
distances[5]-=600;
distances[6]-=200;
}
}
}

leftwheel=BIAS_SPEED;
rightwheel=BIAS_SPEED;
if (gostraight==0) {
for (i=0; i<8; i++) {
leftwheel+=collab_weightleft[i]*(distances[i]>>4);
rightwheel+=collab_weightright[i]*(distances[i]>>4);
}
}

// set robot speed
setSpeed(leftwheel, rightwheel);

if (loopcount<1000) {
loopcount++;
LabPacketReady(&opinionMessage,&i);
} else {
// check if a packet is ready in the com module
if (LabPacketReady(&opinionMessage,&i)) { // i not used but must have 2 params
switch (opinionMessage) {
case LEFT_FOLLOW:
leftPacketCounter++; // it was a LEFT packet so increment left counter
break;
case RIGHT_FOLLOW:
rightPacketCounter++; // it was a RIGHT packet so increment right counter
break;
default:
break;
}
}

// check it it's time to send our own packet
if (HaveToSendOpinion) {
sendLabPacket(opinion,0); // if yes, send it to with our opinion
HaveToSendOpinion = 0; // packet sent, clear the flag
}
}

// make sure the main loop is not done too fast
wait(15000);
}

return 0;
}
```

1.4.9 runfftlistener.{h,c}

```
/*****************************************************************************
*  Programm to demonstrate how FFT works           *
* Version 2.0 aot 2007                           *
* Michael Bonani, Jonathan Besuchet               *
*                                  *
*****************************************************************************/

/*! \file
 * \brief The frequency recognizer using FFT
 * \section sect1 Introduction
 * The runfftlistener programm is made to illustrate how you can use the
 * FFT package of the library. The goal is to determine the frequency of
 * the sound comming from the microphone number 0. If the frequency is
 * under 900Hz the e-puck will turn left. If the frequency is between
 * 900Hz and 1800Hz the e-puck will go forward. If the frequency is over
 * 1800Hz the e-puck will turn right.
 *
 * \section sect2 Playing the demo
 * First of all, move the selector to the position 9 to 15 and reset the e-puck.
 * To play the demo, you have two allternatives:
 * - You are a good whistler. In this case you can drive the e-puck with
```

```
 * your mouse by whistling in the good frequency.
 * - You do not know whistlering. In this case you can play the "sound1.mp3"
 * or "sound2.mp3" which are in the folder "demo_swis_1" to drive the e-puck.
 *
 * \section sect3 Video of the demo
 * - Driving the e-puck by whistlering and playing sound on PC: http://www.youtube.com/watch?v=bfHFo79uZGY
 *
 * \author Jonathan Besuchet
 */

#ifndef _LISTENER
#define _LISTENER

void run_fft_listener();

#endif
```

```
/***************************************************************************

Programm to demonstrate how FFT works
Version 2.0 aot 2007
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \brief The frequency recognizer using FFT
 * \section sect1 Introduction
 * The runfftlistener programm is made to illustrate how you can use the
 * FFT package of the library. The goal is to determine the frequency of
 * the sound comming from the microphone number 0. If the frequency is
 * under 900Hz the e-puck will turn left. If the frequency is between
 * 900Hz and 1800Hz the e-puck will go forward. If the frequency is over
 * 1800Hz the e-puck will turn right.
 *
 * \section sect2 Playing the demo
 * First of all, move the selector to the position 9 to 15 and reset the e-puck.
 * To play the demo, you have two allternatives:
 * - You are a good whistler. In this case you can drive the e-puck with
 * your mouse by whistling in the good frequency.
 * - You do not know whistlering. In this case you can play the "sound1.mp3"
 * or "sound2.mp3" which are in the folder "demo_swis_1" to drive the e-puck.
 *
 * \section sect3 Video of the demo
 * - Driving the e-puck by whistlering and playing sound on PC: http://www.youtube.com/watch?v=bfHFo79uZGY
 *
 * \author Code: Michael Bonani, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

#include <p30Fxxxx.h>
#include <dsp.h>

#include "math.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 "fft/e_fft.h"
#include "fft/e_fft_utilities.h"
#include "motor_led/advance_one_timer/e_motors.h"
#include "motor_led/advance_one_timer/e_agenda.h"
#include "motor_led/advance_one_timer/e_led.h"


/* Extern definitions */
/* Dfinitions externes des variables globales, des diffrents tableaux ou seront stocks les signaux des diffrents micros et la FFT du micro choisi, et
dfinitions de fonctions utiles de moyennage et de copie d'un buffer  l'autre*/
/* Typically, the input signal to an FFT routine is a complex array containing samples of an input signal. */
/* For this example, we will provide the input signal in an array declared in Y-data space. */
extern fractcomplex sigCmpx[FFT_BLOCK_LENGTH] __attribute__ ((section (".ydata, data, ymemory"),aligned (FFT_BLOCK_LENGTH * 2 *2)));
/* Access to the mic. samples */
extern int e_mic_scan[3][FFT_BLOCK_LENGTH];


/*! \brief Localize the bigger pic of the array
 * \param spectre The array in which the FFT was made
 * \param spectre_length The length of the scan in the array
 * \return The index of the bigger pic detected
 */
int localise_pic_max(fractcomplex *spectre, int spectre_length)
{
int i = 0 ;
long ampl_max= 0 ; // Initialisation de l'amplitude maximale
long ampl_courante = 0 ;  //Initialisation de l'amplitude courante
int pic_max;
for (i = 0; i < spectre_length/2; i++)
{
ampl_courante = spectre[i].real*spectre[i].real+spectre[i].imag*spectre[i].imag ; // Calcul de l'amplitude de la FFT  la position i courante

if (ampl_courante>ampl_max) // Si l'amplitude courante est plus grande que l'amplitude maximale mmorise jusqu'ici...
{
pic_max = i ; // La position du Maxima est mmorise dans k_max_1
```

```
ampl_max = ampl_courante ; // La valeur de l'amplitude maximum est remplace par la valeur courante
}
}
return pic_max;
}

/*! \brief Get the max volume of the sound detected
 * \param spectre The array in which the FFT was made
 * \param pic_pos The index of the louder frequency
 * \return The amplitude of the louder frequency
 */
int get_volume(fractcomplex *spectre, int pic_pos)
{
if(pic_pos < 0 || pic_pos >= FFT_BLOCK_LENGTH/2)
return 0;
return spectre[pic_pos].real*spectre[pic_pos].real+spectre[pic_pos].imag*spectre[pic_pos].imag;
}

/*! \brief Display the volume of the soud detected on the LEDs
 * \param volume The max volume of the sound detected
 */
void display_volume_on_led(int volume)
{
e_led_clear();
if(volume > 5 && volume < 20)
e_set_led(4, 1);
else if(volume >= 20 && volume < 50)
{
e_set_led(4, 1);
e_set_led(3, 1);
e_set_led(5, 1);
} else if(volume >= 50 && volume < 200)
{
e_set_led(4, 1);
e_set_led(3, 1);
e_set_led(5, 1);
e_set_led(2, 1);
e_set_led(6, 1);
} else if(volume >= 20 && volume < 300)
{
e_set_led(4, 1);
e_set_led(3, 1);
e_set_led(5, 1);
e_set_led(2, 1);
e_set_led(6, 1);
e_set_led(1, 1);
e_set_led(7, 1);
} else if(volume >= 300)
{
e_set_led(8, 1);
}
}

/*! \brief Calcul the corresponding frequency of the bigger pic detected
 * \param pic_pos The index of the bigger pic detected
 * \return The corresponding frequency
 */
int calcul_frequence(int pic_pos)
{
return (pic_pos*33000)/FFT_BLOCK_LENGTH;
}

/*! \brief Set the speed of the e-puck
 *
 * Set the speed of the e-puck relatively of the frequency
 * of the sound detected.
 * - Turn left if the frequency is under 900Hz
 * - Turn right if the frequency is over 1800Hz
 * - Go forward if the sound is between 900Hz and 1800Hz
 *
 * \param frequency The frequence of the sound
 */
void set_speed(int frequency)
{
if(frequency < 900)
e_set_speed(300, 200);
else if(frequency > 1800)
e_set_speed(300, -200);
else
e_set_speed(400, 0);
}


/*! \brief The "main" function of the demo */
void run_fft_listener(void)
{
int volume;
int pos_pic;
int frequency;

// Initialisations
e_init_port();
e_init_motors();
e_start_agendas_processing();
e_init_ad_scan(MICRO_ONLY);

e_set_speed(400, 0);

while(1)
{
// Scanage des microphone
e_ad_scan_on();

// Attente de l'acquisition de toute les valeurs
while(!e_ad_is_array_filled());
e_ad_scan_off();
```

```
// Centrage du signal au point zro (moyenne = 0)
e_subtract_mean(e_mic_scan[0], FFT_BLOCK_LENGTH, LOG2_BLOCK_LENGTH);

// Copie le signal du micro zero dans le buffer destin  effectuer la FFT
// Affecte les valeurs  sigCmpx.real (rels) et 0  sigCmpx.imag (imaginaires)
e_fast_copy(e_mic_scan[0], (int*)sigCmpx, FFT_BLOCK_LENGTH);

// Le rsultat est sauvegard => On dmarre une nouvelle acquisition
e_ad_scan_on();

// Execution de la FFT sur le buffer
e_doFFT_asm(sigCmpx);

// Recherche de la position K des deux frquences maximum
pos_pic = localise_pic_max(sigCmpx, FFT_BLOCK_LENGTH);

volume = get_volume(sigCmpx, pos_pic);
display_volume_on_led(volume);
if(volume > 100)
{
frequency = calcul_frequence(pos_pic);
set_speed(frequency);
}
}
}
```

1.4.10 runfollowball.{h,c}

```
/************************************************************************
*  Programm to follow balls using the camera       *
* Version 2.0 aot 2007                            *
* Jonathan Besuchet, Alain Balleret               *
*                                 *
************************************************************************/

/*! \file
 * \brief Following balls using the camera
 * \section sect_follow1 Introduction
 * This demo show you how you can drive the camera to make vision based
 * algorithmes.
 * This programm uses the camera to follow balls. There are three differents
 * configurations:
 * - Following the green balls: put the selector on position 6. Now the e-puck
 * will follow the first green ball he saw indifferently of the color.
 * - Following the red balls: put the selector on position 7. Now the e-puck
 * will follow the first red ball he saw indifferently of the color.
 * - Following all the balls: put the selector on position 8. Now the e-puck
 * will follow the first ball he saw indifferently of the color.
 * \image html cam.gif
 *
 * \section sect_follow2 How it works
 * The goal of this programm is to search a ball, when one is found the e-puck
 * has to go in it direction and has to stop at a specified distance. This is
 * done by the following steps:
 * - First of all we have to capture an image. The image is 4 pixels height and
 * 480 pixels width. We take one pixel each for pixels then the image is finally
 * 1 pixel height and 120 pixels width => stored in array 1x120.
 * - We normalize the array to 10 (the mean value of the array is now ten).
 * - We search a pic in the array which correspond of a ball (look at the
 * picture below to see how is the values when a ball is detected).
 * - We work with the motors to do two thinks at the same time: 1) to put this
 * pic on the center of the array => it would place the e-puck across the ball;
 * 2) to adjust the thickness of this pic at the desired value => it would place
 * the e-puck on the good distance of the ball. To have the best control we use
 * a PI (proportionnal intergral) regulator.
 * \subsection pic The graphe of the array when a ball is detected
 * \image html pic.gif
 * \subsection subsect1 Follow the good color
 * When we want to follow all the balls, the camera is used in gray scale mode.
 * \n When we want to follow the green ball for exemple, the camera is used in
 * color mode. In color mode, we have three values: 1 for the red, 1 one
 * for the green, 1 for the blue. Then to follow the green ball we only have to
 * take the red constituent, because the value of green in the red constituent
 * is near zero => it makes a pic in the array.
 * \n When we want to follow the red ball, it's the same principle: we take the
 * green constituent,...
 *
 * \section sect_follow3 Playing the demo
 * The programm works with the contrast of the color between the ball and the
 * environement, so to have the best result you should place the e-puck in an
 * arena which has white wall and good luminosity.
 * \n To play the demo select the configuration you want (follow all, green,
 * red balls) and let's go. The e-puck will turn on himself until he finds a
 * ball.
 *
 * \section sect_follow4 Video of the demo
 * - Following all the balls: http://www.youtube.com/watch?v=Pga7lleqflA
 * - Following the green balls: http://www.youtube.com/watch?v=FGXBy9FOnmw
 *
 * \author Jonathan Besuchet
 */

#ifndef _FOLLOW_BALL
#define _FOLLOW_BALL

void run_follow_ball(void);
void run_follow_ball_green(void);
void run_follow_ball_red(void);

#endif
```

```
/************************************************************************

Programm to follow balls using the camera
Version 2.0 aot 2007
Jonathan Besuchet, Alain Balleret


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Jonathan Besuchet, Alain Balleret

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************/

/*! \file
 * \brief Following balls using the camera
 * \section sect_follow1 Introduction
 * This demo show you how you can drive the camera to make vision based
 * algorithmes.
 * This programm uses the camera to follow balls. There are three differents
 * configurations:
 * - Following the green balls: put the selector on position 6. Now the e-puck
 * will follow the first green ball he saw indifferently of the color.
 * - Following the red balls: put the selector on position 7. Now the e-puck
 * will follow the first red ball he saw indifferently of the color.
 * - Following all the balls: put the selector on position 8. Now the e-puck
 * will follow the first ball he saw indifferently of the color.
 * \image html cam.gif
 *
 * \section sect_follow2 How it works
 * The goal of this programm is to search a ball, when one is found the e-puck
 * has to go in it direction and has to stop at a specified distance. This is
 * done by the following steps:
 * - First of all we have to capture an image. The image is 4 pixels height and
 * 480 pixels width. We take one pixel each for pixels then the image is finally
 * 1 pixel height and 120 pixels width => stored in array 1x120.
 * - We normalize the array to 10 (the mean value of the array is now ten).
 * - We search a pic in the array which correspond of a ball (look at the
 * picture below to see how is the values when a ball is detected).
 * - We work with the motors to do two thinks at the same time: 1) to put this
 * pic on the center of the array => it would place the e-puck across the ball;
 * 2) to adjust the thickness of this pic at the desired value => it would place
 * the e-puck on the good distance of the ball. To have the best control we use
 * a PI (proportionnal intergral) regulator.
 * \subsection pic The graphe of the array when a ball is detected
 * \image html pic.gif
 * \subsection subsect1 Follow the good color
 * When we want to follow all the balls, the camera is used in gray scale mode.
 * \n When we want to follow the green ball for exemple, the camera is used in
 * color mode. In color mode, we have three values: 1 for the red, 1 one
 * for the green, 1 for the blue. Then to follow the green ball we only have to
 * take the red constituent, because the value of green in the red constituent
 * is near zero => it makes a pic in the array.
 * \n When we want to follow the red ball, it's the same principle: we take the
 * green constituent,...
 *
 * \section sect_follow3 Playing the demo
 * The programm works with the contrast of the color between the ball and the
 * environement, so to have the best result you should place the e-puck in an
 * arena which has white wall and good luminosity.
 * \n To play the demo select the configuration you want (follow all, green,
 * red balls) and let's go. The e-puck will turn on himself until he finds a
 * ball.
 *
 * \section sect_follow4 Video of the demo
 * - Following all the balls: http://www.youtube.com/watch?v=Pga71leqf1A
 * - Following the green balls: http://www.youtube.com/watch?v=FGXBy9FOnmw
 *
 * \author Code: Jonathan Besuchet, Alain Balleret \n Doc: Jonathan Besuchet
 */

#include <p30f6014a.h>
#include <string.h>

#include "motor_led/e_epuck_ports.h"
#include "motor_led/e_init_port.h"
#include "uart/e_uart_char.h"
#include "motor_led/advance_one_timer/e_agenda.h"
#include "motor_led/advance_one_timer/e_motors.h"
#include "camera/fast_2_timer/e_po3030k.h"

#include "search_ball.h"

#define NB_VAL 240
#define VIT_ROT_RECHERCHE 300

unsigned char buffer[NB_VAL];

//paramtres de la camra
int epaisseur_ligne_cam = 4;
int pos_ligne1 = ARRAY_WIDTH/2-4/2;

void select_cam_mode(int mode);
void execute(unsigned char *buf_execute, Epuck *epuck);
void follow_red(unsigned char *buf, int size);
void follow_green(unsigned char *buf, int size);

/*------ internal functions ------*/

/*! \brief Set the mode of the camera
 * \param mode Put RGB_565_MODE or GREY_SCALE_MODE
 */
```

```
void select_cam_mode(int mode)
{
e_po3030k_config_cam(pos_ligne1,0, epaisseur_ligne_cam,ARRAY_HEIGHT, 4,4,  mode);
e_po3030k_set_mirror(1,1);
e_po3030k_write_cam_registers();
}

/*! \brief Execute all the steps to follow the ball
 *
 * The steps are:
 * - normalizing the array to 10
 * - searching the ball (the pic an the array)
 * - if a ball is found => follow him
 * \param buf_execute the array containing the image
 * \param epuck The struct which contains the e-puck state
 */
void execute(unsigned char *buf_execute, Epuck *epuck)
{
char pic_found;

normalize(buf_execute, NB_VAL/2);
pic_found = search_ball(epuck, buf_execute, NB_VAL/2); // fonction permettant de trouver un pic

if(pic_found == PIC_FOUND)
{
if(epuck->state == IS_SEARCHING_BALL) {
ARW();
}
epuck->state = IS_FOLLOWING_BALL; // change l'tat de l'epuck dans la structure
BODY_LED = 1;
goto_ball(epuck);
}
else
{
ARW();
epuck->state = IS_SEARCHING_BALL;
BODY_LED = 0;

//lorsqu'il perd la balle de vue, le epuck tourne sur lui-meme
//dans le sens que prenait la balle.
epuck->lin_speed = 0;

//tourne dans la mme direction que prenait la balle
//lorsqu'il l'a perdue.
if(epuck->angle_ball > 0)
{
e_set_speed_left(VIT_ROT_RECHERCHE);
e_set_speed_right(-VIT_ROT_RECHERCHE);
}
else
{
e_set_speed_left(-VIT_ROT_RECHERCHE);
e_set_speed_right(VIT_ROT_RECHERCHE);
}
}
}

/*! \brief The filter to follow the red ball
 * \param buf The array containing the image
 * \param size The size of the array
 */
void follow_red(unsigned char *buf, int size)
{
int i;
unsigned char green;
for(i=0; i<size/2; i++)
{
green = (((buf[2*i] & 0x07) << 5) | ((buf[2*i+1] & 0xE0) >> 3));
//blue = ((buf[2*i+1] & 0x1F) << 3);
buf[i] = green;
}
}

/*! \brief The filter to follow the green ball
 * \param buf The array containing the image
 * \param size The size of the array
 */
void follow_green(unsigned char *buf, int size)
{
int i;
unsigned char red;
for(i=0; i<size/2; i++)
{
red = (buf[2*i] & 0xF8);
//blue = ((buf[2*i+1] & 0x1F) << 3);
buf[i] = red;
}
}

/*------ external functions ------*/

/*! \brief The "main" function to follow all the balls */
void run_follow_ball(void)
{
unsigned char *tab_start = buffer;
unsigned char *tab_middle = buffer + NB_VAL/2;

Epuck epuck;

epuck_init(&epuck);

e_init_port();    // configure port pins
e_start_agendas_processing();
e_init_motors();
e_init_uart1();   // initialize UART to 115200 Kbaud

e_po3030k_init_cam();
```

```
select_cam_mode(GREY_SCALE_MODE);

while(1)
{
//*************************************************
// on rempli le debut du buffer et on travaille sur la fin

e_po3030k_launch_capture((char *)tab_start);

execute(tab_middle, &epuck);

while(!e_po3030k_is_img_ready());


//*************************************************
// on rempli la fin du buffer et on travaille sur le debut

e_po3030k_launch_capture((char *)tab_middle);

execute(tab_start, &epuck);

while(!e_po3030k_is_img_ready());
}
}

/*! \brief The "main" function to follow the green ball */
void run_follow_ball_green(void)
{
unsigned char *tab_start = buffer;
unsigned char *tab_middle = buffer + NB_VAL/2;

Epuck epuck;

epuck_init(&epuck);

e_init_port();    // configure port pins
e_start_agendas_processing();
e_init_motors();
e_init_uart1();   // initialize UART to 115200 Kbaud

e_po3030k_init_cam();
select_cam_mode(RGB_565_MODE);

while(1)
{
//*************************************************
// on rempli tous le buffer et on travaille sur le debut

e_po3030k_launch_capture((char *)tab_start);
while(!e_po3030k_is_img_ready());

follow_green(tab_start, NB_VAL);
execute(tab_start, &epuck);
}
}

/*! \brief The "main" function to follow the red ball */
void run_follow_ball_red(void)
{
unsigned char *tab_start = buffer;
unsigned char *tab_middle = buffer + NB_VAL/2;

Epuck epuck;

epuck_init(&epuck);

e_init_port();    // configure port pins
e_start_agendas_processing();
e_init_motors();
e_init_uart1();   // initialize UART to 115200 Kbaud

e_po3030k_init_cam();
select_cam_mode(RGB_565_MODE);

while(1)
{
//*************************************************
// on rempli tous le buffer et on travaille sur le debut

e_po3030k_launch_capture((char *)tab_start);
while(!e_po3030k_is_img_ready());

follow_red(tab_start, NB_VAL);
execute(tab_start, &epuck);
}
}
```

1.4.11 rungrounddirection.{h,c}

```
/************************************************************************
*  Programm to show the gravity direction       *
* Version 3.0 7 janvier 2005                          *
* Lucas Meier               *
*                                  *
************************************************************************/

/*! \file
 * \brief Showing the ground direction
 * \section sect_dir1 Introduction
 * This programm works with the accelerometer to show you the ground
 * direction by turning on the appropriate LED.
 *
 * \section sect_dir2 Playing the demo
 * First of all, move the selector to the position 3 and reset the e-puck.
 * Now take the e-puck in the hand and turn it with the Z axis horizontal.
```

```
 *
 * \section sect_dir3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=a_4eymEv4bs
 *
 * \author Jonathan Besuchet
 */


#ifndef _GROUND_DIR
#define _GROUND_DIR

void run_grounddirection();

#endif
```

```
/*****************************************************************************

Programm to show the gravity direction
Version 3.0 7 janvier 2005
Lucas Meier


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \brief Showing the ground direction
 * \section sect_dir1 Introduction
 * This programm works with the accelerometer to show you the ground
 * direction by turning on the appropriate LED.
 *
 * \section sect_dir2 Playing the demo
 * First of all, move the selector to the position 3 and reset the e-puck.
 * Now take the e-puck in the hand and turn it with the Z axis horizontal.
 *
 * \section sect_dir3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=a_4eymEv4bs
 *
 * \author Code: Lucas Meier \n Doc:  Jonathan Besuchet
 */

#include "math.h"
#include "stdlib.h"

#include "a_d/advance_ad_scan/e_ad_conv.h"
#include "a_d/advance_ad_scan/e_acc.h"
#include "motor_led/e_epuck_ports.h"

#include "rungrounddirection.h"

/*! \brief The "main" function of the demo */
void run_grounddirection(void)
{
e_init_port();
    e_init_ad_scan(ALL_ADC);
e_acc_calibr();

while(1)
{
e_display_angle();
}
}
```

1.4.12 runlocatesound.{h,c}

```
/*****************************************************************************
*  Programm which locate the direction of the sound       *
* Version 1.0 aot 2007                                    *
* Michael Bonani, Jonathan Besuchet                        *
*                                    *
*****************************************************************************/

/*! \file
 * \brief Locate the direction of the sound
 * \section sect1 Introduction
 * This demo uses the three microphones to locate the direction of a sound.
 * \n To determine the direction of the sound we use the fact that the sound
 * is detected by each microphone on different time. As we know the sound
 * speed, we can determine from where come the sound.
 *
 * \section sect_sound2 Playing the demo
 * First of all, move the selector to the position 1 and reset the e-puck.
 * Clap your hands and the e-puck will turn on the LED which correspond to
 * the direction of the sound (one second). Then the e-puck turn on himslef
 * and look in the direction where the sound was made.
 *
 * \section sect_sound3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=1RRqZMmfrfI
 *
 * \author Jonathan Besuchet
 */
#ifndef _LOCATES
```

```
#define _LOCATES

void run_locatesound();

#endif
```

```
/*****************************************************************************

Programm which locate the direction of the sound
Version 1.0 aot 2007
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \brief Locate the direction of the sound
 * \section sect1 Introduction
 * This demo uses the three microphones to locate the direction of a sound.
 * \n To determine the direction of the sound we use the fact that the sound
 * is detected by each microphone on different time. As we know the sound
 * speed, we can determine from where come the sound.
 *
 * \section sect_sound2 Playing the demo
 * First of all, move the selector to the position 1 and reset the e-puck.
 * Clap your hands and the e-puck will turn on the LED which correspond to
 * the direction of the sound (one second). Then the e-puck turn on himslef
 * and look in the direction where the sound was made.
 *
 * \section sect_sound3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=1RRqZMmfrfI
 *
 * \author Code: Michael Bonani, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

#include "p30f6014A.h"
#include <math.h>

#include "motor_led/advance_one_timer/e_motors.h"
#include "a_d/advance_ad_scan/e_ad_conv.h"
#include "a_d/advance_ad_scan/e_micro.h"

#include "runlocatesound.h"

#define PI 3.1415

/* defines used in main.c                          */
#define MEAN_MAX 100 // Length of the mean_table
#define MEAN_MAX_INV 0.01 // Inversed value of MEAN_MAX
#define PERCENT 0.1 // Defines range to distinguish between noise an signal

/* defines used in ad_conv_int.c */
//#define MIC_SAMP_NB 100 // number of microphon samples to store

/* defines used in find_direction */
//maximum_delta_t =
//sampling_frequency[Hz] * distance_between_microphones[m] / speed_of_sound[m/s];
#define MAXIMUM_DELTA_T1 6.
#define MAXIMUM_DELTA_T2 5.

/* defines used in find_delta_t.c */
#define TAU_RANGE 14 // Needs to be a pair number

/* defines used in turn_to_direction.c */
#define TURN_SPEED 1000
#define STEPS_FOR_2PI 1300.

extern int e_mic_scan[3][MIC_SAMP_NB]; //Array to store the mic values
extern unsigned int e_last_mic_scan_id; //ID of the last scan in the mic array
extern unsigned char is_ad_acquisition_completed; //to check if the acquisition is done

//int new_sample;
float mean_table[3][MEAN_MAX];
int mean_nb;
float mean[3];
float signal_max[3], signal_min[3];


void calculate_average(int *);
void init_micro(void);
void record_sound(void);
int check_for_event(void);
void filter_signal(void);
void calculate_average(int *current_sample);
//int get_micro (unsigned int micro_ID);
int get_micro_average(unsigned int micro_ID, unsigned int filter_size);
float calculate_direction(void);
int find_delta_t(int mic1_nb,int mic2_nb);
void show_led(float angle);
void turn_to_direction(float direction);


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
// init sound
//
// Initialize everything necessary to record sound
//
// in:  void
// out: void
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void init_micro(void)
{
int j, k;

mean_nb = 0; // start to fill up the table from position 0
while(!e_ad_is_acquisition_completed());  // wait until a new sample has been taken

// initialize the table of average values
// save signal level
for (k=0; k<3; k++)
{
mean[k]=(float)e_get_micro((unsigned) k);     // start at a given average value
}

    for( k=0; k<3; k++)
    {
for (j=0; j<MEAN_MAX; j++)      // fill the mean_table with predefined values
{
mean_table[k][j]= mean[k] * MEAN_MAX_INV;
}

signal_min[k] = mean[k] - PERCENT * mean[k]; // predefine level for eventdetecting
signal_max[k] = mean[k] + PERCENT * mean[k];
}
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// record sound
//
// Fills up the memory with the recorded sound from all
// three microphones at a sampling rate of about 25kHz
//
// in:  void
// out: void
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void record_sound(void)
{
e_last_mic_scan_id = 0;   // reset the scan_ID to 0 so that we have the full table with the sound
while(!e_ad_is_acquisition_completed()); // e_last_mic_scan_id <= (MIC_SAMP_NB - 2) ); //wait until the table is full
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// check for event
//
// Checks if an event has happened. This function dynamically
// takes the average of the soundstream and then checks, whether
// it is tresspassing a predefined treshold.
//
// in:  void
// out: int (1: no event has occured)
//          (0: an event has occured!)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
int check_for_event(void)
{
int not_event;
int current_sample[3];

// get one single sample for all 3 microphones
current_sample[0]=e_get_micro((unsigned) 0);
current_sample[1]=e_get_micro((unsigned) 1);
current_sample[2]=e_get_micro((unsigned) 2);

    // Detect event on any of the 3 microphones
not_event = (  ((current_sample[0]<signal_max[0]) && (current_sample[0]>signal_min[0])) ||
   ((current_sample[1]<signal_max[1]) && (current_sample[1]>signal_min[1])) ||
   ((current_sample[2]<signal_max[2]) && (current_sample[2]>signal_min[2])) );

calculate_average(current_sample); // dynamically calculates the new average value of the noise

return not_event;    // if no event, return 1
                     // if event, return 0
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// filter signal
//
// Filters the signal, so that the detect_direction module
// has an optimum signal.
// This includes shifting the signal around zero and then
// takes the absolute value.
//
// in:  void
// out: void
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void filter_signal(void)
{
int i, k;

for (i=0; i<3; i++)   // for all three mics
{
for (k = 0; k < MIC_SAMP_NB; k++) // for the whole signal
{
e_mic_scan[i][k] -= mean[i]; // shift the signal down to around 0
if (e_mic_scan[i][k] < 0)         // take the absolute value
e_mic_scan[i][k] = -e_mic_scan[i][k];   // --> gives better values in the cross-correlation
}
}
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
//
// Private functions
//
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// calculate average
//
// Dynamically calculates the average of the sound stream.
// This is necessary because the average signal coming from the
// ADC is not always the same for all 3 microphones and tests
// have shown that it may shift during the exexution of the program
//
// in:  pointer to the current sample
// out: void
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void calculate_average(int *current_sample)
{
int k;

while(!e_ad_is_acquisition_completed());  // wait until a new sample has been taken

for( k=0; k<3; k++) // for all three mics calculate average
    {
mean[k] = mean[k] - mean_table[k][mean_nb];
mean_table[k][mean_nb] = MEAN_MAX_INV * (float)current_sample[k];
mean[k] += mean_table[k][mean_nb];

// adapt treshold to detect an event
signal_min[k] = mean[k] - PERCENT * mean[k];
signal_max[k] = mean[k] + PERCENT * mean[k];
}

// ensure a circular memory usage
if (mean_nb<MEAN_MAX-1)
mean_nb++;
else
mean_nb = 0;
}

/*****************************************************************
 * Get the average on a given number of sample from a micro
 *
 * @param micro_ID IN micro's ID (0, 1, or 2)
 * (use MICR0, MICR1, MICR2 defined in ad_conv_int.h)
 * @param filter_size IN number of sample to average
 * @return result OUT last value of the micro
 *****************************************************************/
int get_micro_average(unsigned int micro_ID, unsigned int filter_size)
{
long temp = 0;
int i,j;

// channel ID must be between 0 to 2 and
// filter_size must be between 1 to SAMPLE_NUMBER
if ((micro_ID < 3) &&
(filter_size > 0) && (filter_size <= MIC_SAMP_NB))
{
for (i=0, j=(MIC_SAMP_NB-(filter_size-(e_last_mic_scan_id+1)))%MIC_SAMP_NB ; i<filter_size ; i++, j=(j+1)%MIC_SAMP_NB)
{
temp += e_mic_scan[micro_ID][j];
}
}
return ((int)(temp/filter_size));
}

float calculate_direction(void)
{
int delta_t1, delta_t2;
float direction, angle1, angle2;

// first get the phase-shift between the right and the left microphone
delta_t1 = find_delta_t(0,1);

// calculate the angle (between -90 and +90 where the sound is coming from)
if (delta_t1 >= MAXIMUM_DELTA_T1)
angle1 = PI * 0.5;  // to avoid NaN of asin
else if (delta_t1 <= -MAXIMUM_DELTA_T1)
angle1 = -PI * 0.5;  // to avoid NaN of asin
else
angle1 = asin( (float)(delta_t1)/MAXIMUM_DELTA_T1 );


// now if the signal is coming from the right, we check the phase-shift between the
// left and the rear microphone in order to find out if the direction is
// angle1 or (180 - angle1)
// if thes signal coming from the left, we make the same test with the right and the
// rear microphone
if(angle1 > 0)
{
delta_t2 = find_delta_t(2,1);   // phase shift between left and rear microphone

if (delta_t2 >= MAXIMUM_DELTA_T2)
angle2 = PI * 0.5;  // to avoid NAN of asin
else if (delta_t2 <= -MAXIMUM_DELTA_T2)
angle2 =-PI * 0.5;  // to avoid NAN of asin
else
angle2 = asin( (float)delta_t2/MAXIMUM_DELTA_T2 );

if(angle2 > PI/6.)   // if the second angle is bigger than +30
direction = PI - angle1;    // the direction = 90-angle1
else direction = angle1;
}
else
{
delta_t2 = find_delta_t(0,2);  // phase shift between right and rear microphone
```

```
if (delta_t2 >= MAXIMUM_DELTA_T2)
angle2 = PI * 0.5;  // to avoid NAN of asin
else if (delta_t2 <= -MAXIMUM_DELTA_T2)
angle2 = -PI * 0.5;  // to avoid NAN of asin
else
angle2 = asin( (float)delta_t2/MAXIMUM_DELTA_T2 );

if(angle2 < -PI/6.) // if the second angle is smaller than -30
direction = PI - angle1; // the direction = 90-angle1
else direction = angle1;
}

// We want an angle strictly between [0,2*PI]
if (direction < 0)
direction = 2*PI + direction;

return direction;
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// find delta t
//
// Finds the phase-shift between two signal.
// Basically, this function finds the maximum of the
// cross-correlation between two signals.
//
// in:  int (microphone number of signal 1)
//      int (microphone number of signal 2)
// out: int (time expressed as number of samples taken)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

int find_delta_t(int mic1_nb,int mic2_nb)
{
int delta_t, tau, k;
long int correlation, max;
//extern int mic_scan[3][MIC_SAMP_NB];

int tau_min = -TAU_RANGE / 2;
int tau_max = TAU_RANGE / 2;

int save_sound_start = TAU_RANGE / 2 + 1;
int save_sound_end = MIC_SAMP_NB - TAU_RANGE / 2 - 1;


max = 0;

for (tau = tau_min; tau < tau_max; tau++)
{
//reset the the correlation value
correlation = 0;

    // For each tau calculate the correlation between the two signals
    for (k = save_sound_start; k < save_sound_end; k++)
    {
    correlation += (long int)(e_mic_scan[mic1_nb][k]) * (long int)(e_mic_scan[mic2_nb][k+tau] );
}

// find out if this correlation is the biggest one so far. --> If yes,
     // save the value of tau --> This gives us the phaseshift between the
    // signals
        if (correlation > max)
        {
        max = correlation;
        delta_t = tau;
    }

}

return delta_t;
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// show led
//
// Lights up the LED in the appropriate direction
//
// in:  float (angle between 0 and 2PI clockwise)
// out: void
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

void show_led(float angle)
{
int led_nb = 6;
long int i;

// led_nb corresponds to the appropriate bit number in the LATA register
if ( angle > (PI/8) )
led_nb = 7;
if ( angle > (3*PI/8) )
led_nb = 9;
if ( angle > (5*PI/8) )
led_nb = 12;
if ( angle > (7*PI/8) )
led_nb = 10;
if ( angle > (9*PI/8) )
led_nb = 13;
if ( angle > (11*PI/8) )
led_nb = 14;
if ( angle > (13*PI/8) )
led_nb = 15;
if ( angle > (15*PI/8) )
led_nb = 6;

// set the bit on PortA to illuminate the led
LATA = 1 << led_nb;
```

```
for (i=0;i<200000;i++);   // Wait to indicate the direction

LATA = 0; // turn all LEDs off
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// turn to direction
//
// Turns the robot to the appropriate direction
//
// in:  float (angle between 0 and 2PI clockwise)
// out: void
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

void turn_to_direction(float direction)
{
int end_turn;

if (direction < PI)     // turn clockwise
{
e_set_steps_left(0);
e_set_speed_left(TURN_SPEED);  // motor speed in steps/s
e_set_speed_right(-TURN_SPEED);  // motor speed in steps/s

// calculate how many steps the robot needs to turn
end_turn = (int)(STEPS_FOR_2PI*(direction/(2*PI)));
while(e_get_steps_left() < end_turn);   // turn until done
}
else  // turn counterclockwise
{
e_set_steps_right(0);
e_set_speed_left(-TURN_SPEED);  // motor speed in steps/s
e_set_speed_right(TURN_SPEED);  // motor speed in steps/s

// calculate how many steps the robot needs to turn
end_turn = (int)(STEPS_FOR_2PI*((2*PI-direction)/(2*PI)));
while(e_get_steps_right() < end_turn);   // turn until done
}

// stop motors
e_set_speed_left(0);  // motor speed in steps/s
e_set_speed_right(0);  // motor speed in steps/s
}

void run_locatesound() {
int not_event;
float direction;

e_init_ad_scan(MICRO_ONLY);     // initialize and start ad_conversion
    e_init_motors();
    e_start_agendas_processing();

init_micro();

while(1) {
init_micro();
not_event = 1;
while (not_event)    // do this loop until an event has occured
{
not_event = check_for_event();
}
// if there is an event then:
record_sound();         // fill up the memory with the claping sound

e_ad_scan_off(); // disable AD conversion (avoid interferes with the calculations)

filter_signal();        // filters the signals

direction = calculate_direction();   // do all the calculations where the sound is coming from

e_ad_scan_on(); // enable AD conversion

show_led(direction);    // indicate where the sound is coming from

turn_to_direction(direction); // turn the robot to the direction the sound is coming from
}

}
```

1.4.13 runwallfollow.{h,c}

```
/***************************************************************************
*  Programm to follow a wall       *
* Version 1.0 aot 2007                          *
* Michael Bonani, Jonathan Besuchet               *
*                                    *
***************************************************************************/

/*! \file
 * \brief Follow a wall
 * \section sect1 Introduction
 * With this program, the e-puck will follow a wall.
 *
 * \section sect_sound2 Playing the demo
 * First of all, move the selector to the position 2 and reset the e-puck.
 * \n The e-puck will now follow the first wall he finds. You can change
 * the side on which the e-puck must follow the wall with the selector.
 *
 * \section sect_sound3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=xaqpoQ_XGbU
 *
 * \author Jonathan Besuchet
 */

#ifndef _WALLFOLLOW
```

```
#define _WALLFOLLOW


void run_wallfollow();

#endif
```

```
/*****************************************************************************

Programm to follow a wall
Version 1.0 aot 2007
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \brief Follow a wall
 * \section sect1 Introduction
 * With this program, the e-puck will follow a wall.
 *
 * \section sect_sound2 Playing the demo
 * First of all, move the selector to the position 2 and reset the e-puck.
 * \n The e-puck will now follow the first wall he finds. You can change
 * the side on which the e-puck must follow the wall with the selector.
 *
 * \section sect_sound3 Video of the demo
 * The video of this demo: http://www.youtube.com/watch?v=xaqpoQ_XGbU
 *
 * \author Code: Michael Bonani, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

#include "p30f6014A.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "motor_led/e_epuck_ports.h"
#include "motor_led/e_init_port.h"
#include "motor_led/advance_one_timer/e_agenda.h"
#include "motor_led/advance_one_timer/e_led.h"
#include "motor_led/advance_one_timer/e_motors.h"
#include "uart/e_uart_char.h"
#include "a_d/advance_ad_scan/e_ad_conv.h"
#include "a_d/advance_ad_scan/e_prox.h"

#include "utility.h"
#include "runwallfollow.h"


#define LEFT_FOLLOW 0 // behaviors IDs
#define RIGHT_FOLLOW 1

#define NB_SENSORS          8 // number of sensors
#define BIAS_SPEED       350 // robot bias speed
#define SENSOR_THRESHOLD 300 // discount sensor noise below threshold
#define MAXSPEED  800 // maximum robot speed


int follow_sensorzero[8];
int follow_weightleft[8] = {-10,-10,-5,0,0,5,10,10};
int follow_weightright[8] = {10,10,5,0,0,-5,-10,-10};

/*! \breif Callibrate the proxymity sensor */
/*void follow_sensor_calibrate() {
int i, j;
char buffer[80];
long sensor[8];

for (i=0; i<8; i++) {
sensor[i]=0;
}

for (j=0; j<32; j++) {
for (i=0; i<8; i++) {
sensor[i]+=e_get_prox(i);
}
wait(10000);
}

for (i=0; i<8; i++) {
follow_sensorzero[i]=(sensor[i]>>5);
sprintf(buffer, "%d, ", follow_sensorzero[i]);
e_send_uart1_char(buffer, strlen(buffer));
}

sprintf(buffer, " calibration done\r\n");
e_send_uart1_char(buffer, strlen(buffer));
wait(100000);
}*/

/*! \brief Looking at the selector value
 * \return The selector value from 0 to 15
 */
```

```
int followgetSelectorValue() {
return SELECTOR0 + 2*SELECTOR1 + 4*SELECTOR2 + 8*SELECTOR3;
}


/*! \breif Read the sensors proxymities
 * \param sensorTable Where the value must be stocked
 */
void followGetSensorValues(int *sensorTable) {
unsigned int i;
for (i=0; i < NB_SENSORS; i++) {
sensorTable[i] = e_get_calibrated_prox(i); //e_get_prox(i) - follow_sensorzero[i];
}
}

/*! \brief Set robot speed */
void followsetSpeed(int LeftSpeed, int RightSpeed) {
if (LeftSpeed < -MAXSPEED) {LeftSpeed = -MAXSPEED;}
if (LeftSpeed >  MAXSPEED) {LeftSpeed =  MAXSPEED;}
if (RightSpeed < -MAXSPEED) {RightSpeed = -MAXSPEED;}
if (RightSpeed >  MAXSPEED) {RightSpeed =  MAXSPEED;}
e_set_speed_left(LeftSpeed);
e_set_speed_right(RightSpeed);
}

/*! \brief The "main" function of the program */
void run_wallfollow() {
int leftwheel, rightwheel; // motor speed left and right
int distances[NB_SENSORS]; // array keeping the distance sensor readings
int i; // FOR-loop counters
int gostraight;
int loopcount;
unsigned char selector_change;

e_init_port();
e_init_ad_scan(ALL_ADC);
e_init_motors();
e_start_agendas_processing();

//follow_sensor_calibrate();

e_activate_agenda(left_led, 2500);
e_activate_agenda(right_led, 2500);
e_pause_agenda(left_led);
e_pause_agenda(right_led);

e_calibrate_ir();
loopcount=0;
selector_change = !(followgetSelectorValue() & 0x0001);

while (1) {
followGetSensorValues(distances); // read sensor values

gostraight=0;
if ((followgetSelectorValue() & 0x0001) == RIGHT_FOLLOW) {
if(selector_change == LEFT_FOLLOW) {
selector_change = RIGHT_FOLLOW;
e_led_clear();
e_pause_agenda(left_led);
e_restart_agenda(right_led);
}
for (i=0; i<8; i++) {
if (distances[i]>50) {break;}
}
if (i==8) {
gostraight=1;
} else {
follow_weightleft[0]=-10;
follow_weightleft[7]=-10;
follow_weightright[0]=10;
follow_weightright[7]=10;
if (distances[2]>300) {
distances[1]-=200;
distances[2]-=600;
distances[3]-=100;
}
}
} else {
if(selector_change == RIGHT_FOLLOW) {
selector_change = LEFT_FOLLOW;
e_led_clear();
e_pause_agenda(right_led);
e_restart_agenda(left_led);
}
for (i=0; i<8; i++) {
if (distances[i]>50) {break;}
}
if (i==8) {
gostraight=1;
} else {
follow_weightleft[0]=10;
follow_weightleft[7]=10;
follow_weightright[0]=-10;
follow_weightright[7]=-10;
if (distances[5]>300) {
distances[4]-=100;
distances[5]-=600;
distances[6]-=200;
}
}
}

leftwheel=BIAS_SPEED;
rightwheel=BIAS_SPEED;
if (gostraight==0) {
for (i=0; i<8; i++) {
leftwheel+=follow_weightleft[i]*(distances[i]>>4);
```

```
rightwheel+=follow_weightright[i]*(distances[i]>>4);
}
}

// set robot speed
followsetSpeed(leftwheel, rightwheel);

wait(15000);
}
}
```

1.4.14 search_ball.{h,c}

```
//*****************************************
//CONSTANTES
#define PIC_FOUND 1
#define PIC_NOT_FOUND -1
#define IS_SEARCHING_BALL 0
#define IS_FOLLOWING_BALL 1

//*****************************************
//structure dfinissant l'epuck
#ifndef Epuck
typedef struct
{
   char state;
int dist_ball;
int angle_ball;
int lin_speed;
int angle_speed;
} Epuck;
#endif

//*****************************************
//fcts accessible de l'exterieur de search_ball
void epuck_init(Epuck *epuck);

void normalize(unsigned char buffer[], int nb_val);

int search_ball(Epuck *epuck, unsigned char buffer[], int nb_val);

void goto_ball(Epuck *epuck);

void ARW();
```

```
/*******************************************************************************

Code to search a ball around the e-puck
December 2007: first version
Jonathan Besuchet & Alain Balleret


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Jonathan Besuchet & Alain Balleret

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*******************************************************************************/

#include <p30f6014a.h>
#include <stdlib.h>

#include "search_ball.h"
#include "./../../library/motor_led/advance_one_timer/e_motors.h"

//*****************************************************
//variable globale
#define PIC_SIZE_MIN 3
static float ui_lin = 0.0;

//*****************************************************
//declaration des fcts interne au module search_ball

int get_tab_moy(unsigned char tab[], int start, int end);
int calc_pic_datas_g(int *largeur_pic, int *pos_pic, unsigned char buffer[], int nb_val);
int calc_pic_datas_d(int *largeur_pic, int *pos_pic, unsigned char buffer[], int nb_val);
int calc_lin_speed(int distance, int gain);
int calc_angle_speed(int pos_pic, int gain);


//fonction permettant de faire la moyenne d'un tableau
int get_tab_moy(unsigned char tab[], int start, int end)
{
int i;
int moy = 0;
if(start == end)
return tab[start];
for(i=start; i<end; i++)
{
moy += tab[i];
}
if(moy == 0)
return 0;
return moy/(end-start);
}
```

```
// recherche de pic en partant de la gauche, c'est  dire de la premire valeur du tableau
int calc_pic_datas_g(int *largeur_pic, int *pos_center_pic, unsigned char buffer[], int nb_val)
{
static int nb_moy = 10;
int pic1, pic2;
int difference;

//dtection de l'endroit du premier pic en moyennant les 10 dernire valeur et
//en les comparant au pixel en cours (on suppose que le pic est aprs les 10
//premiers 1ments) le flanc du  pic doit etre superieur a 3.
pic1 = nb_moy+1;
difference = 0;
// tant que la diffrence est inf  3, et que toutes les valeur n'ont pas t test
// recherche d'un pic descandant
while(pic1 < nb_val-1)
{
difference = get_tab_moy(buffer, pic1-nb_moy-1, pic1-1) - ((int)buffer[pic1]+(int)buffer[pic1+1]) / 2;
if(difference > PIC_SIZE_MIN)
break;
pic1++;
}

//determination de la largeur du pic.
    //si no_pt = nb_val ca veut dire qu'il ny a pas de pic.
if(pic1 >= nb_val || difference <= PIC_SIZE_MIN)
return PIC_NOT_FOUND;

pic2 = pic1 + 1;
difference = 0;
// tant que la diffrence est inf  3, et que toutes les valeur n'ont pas t test
// recherche d'un pic montant
while(pic2 < nb_val-1)
{
difference = ((int)buffer[pic2]+(int)buffer[pic2]) / 2 - get_tab_moy(buffer, pic1, pic2);
if(difference > PIC_SIZE_MIN)
break;
pic2++;
}

*largeur_pic = pic2-pic1;
// on calcule la position du centre du pic par rapport au centre de la camera.
if(pic2 >= nb_val)
*pos_center_pic = nb_val/2;
else
*pos_center_pic = pic1 + (pic2-pic1)/2 - nb_val/2;
return PIC_FOUND;
}

// recherche de pic en partant de la droite, c'est  dire de la dernire valeur du tableau
int calc_pic_datas_d(int *largeur_pic, int *pos_center_pic, unsigned char buffer[], int nb_val)
{
static int nb_moy = 10;
int pic1, pic2;
int difference;

//dtection de l'endroit du premier pic en moyennant les 10 dernire valeur et
//en les comparant au pixel en cours (on suppose que le pic est aprs les 10
//premiers 1ments) le flanc du  pic doit etre superieur a 3.
pic1 = nb_val - (nb_moy+1);
difference = 0;
// tant que la diffrence est inf  3, et que toutes les valeur n'ont pas t test
// recherche d'un pic descandant
while(pic1 > 0)
{
difference = get_tab_moy(buffer,  pic1+1, pic1+nb_moy+1) - ((int)buffer[pic1]+(int)buffer[pic1-1]) / 2;
if(difference > PIC_SIZE_MIN)
break;
pic1--;
}

//determination de la largeur du pic.
    //si no_pt = 0 ca veut dire qu'il ny a pas de pic.
if(pic1 == 0 || difference <= PIC_SIZE_MIN)
return PIC_NOT_FOUND;

pic2 = pic1 - 1;
difference = 0;
// tant que la diffrence est inf  3, et que toutes les valeur n'ont pas t test
// recherche d'un pic montant
while(pic2 > 0)
{
difference = ((int)buffer[pic2]+(int)buffer[pic2-1]) / 2 - get_tab_moy(buffer, pic2+1, pic1);
if(difference > PIC_SIZE_MIN)
break;
pic2--;
}

// on calcule la position du centre du pic par rapport au centre de la camera.
*largeur_pic = pic1-pic2;
if(pic2 <= 0)
*pos_center_pic = -nb_val/2;
else
*pos_center_pic = pic2 + (pic1-pic2)/2 - nb_val/2;
return PIC_FOUND;
}

//*********************************************************************
//Fonctions externe

void epuck_init(Epuck *epuck)
{
epuck->state = IS_SEARCHING_BALL; // initialise l'tat de l'epuck en recherche de balle
epuck->dist_ball = -1;
epuck->angle_ball = -1;
epuck->lin_speed = 0;
epuck->angle_speed = 300;
}
```

```
// permet de normaliser le buffer
void normalize(unsigned char buffer[], int nb_val)
{
int moy = get_tab_moy(buffer, 0, nb_val);
int i;

if(moy == 0)
return;
for(i=0; i<nb_val; i++)
{
buffer[i] = (10*buffer[i])/moy;
}
}


//Recherche et retourne la position et l'paisseur du pic.
int search_ball(Epuck *epuck, unsigned char buffer[], int nb_val)
{
int largeur_pic_g, largeur_pic_d;
int pos_pic_g, pos_pic_d;
char pic_found_g, pic_found_d;

//on fait une fois la recherche du pic depuis la gauche 'g'
//et une fois depuis la droite 'd'.
pic_found_g = calc_pic_datas_g(&largeur_pic_g, &pos_pic_g, buffer, nb_val);
pic_found_d = calc_pic_datas_d(&largeur_pic_d, &pos_pic_d, buffer, nb_val);

//on retoune la largeur du pic. PIC_NOT_FOUND si pas de pic trouv
if(pic_found_g == PIC_NOT_FOUND && pic_found_d == PIC_NOT_FOUND)
return PIC_NOT_FOUND;
//si seul l'acquisition par la gauche retourne une valeur
else if(pic_found_g == PIC_FOUND && pic_found_d == PIC_NOT_FOUND)
{
epuck->dist_ball = largeur_pic_g;
epuck->angle_ball = pos_pic_g;
return PIC_FOUND;
}
//si seul l'acquisition par la droite retourne une valeur
else if(pic_found_g == PIC_NOT_FOUND && pic_found_d == PIC_FOUND)
{
epuck->dist_ball = largeur_pic_d;
epuck->angle_ball = pos_pic_d;
return PIC_FOUND;
}
// si les deux acquisitions rendent une valeur, alors on fait la moyenne des deux
else
{
epuck->dist_ball = (largeur_pic_g + largeur_pic_d) / 2;
epuck->angle_ball = (pos_pic_g + pos_pic_d) / 2;
return PIC_FOUND;
}
}

//fonction qui envoie l'epuck vers la balle
void goto_ball(Epuck *epuck)
{
int lin_speed = 0;
int angle_speed = 0;
int gain_lin = 35;
int gain_angle = 6;

lin_speed = calc_lin_speed(epuck->dist_ball, gain_lin);
angle_speed = calc_angle_speed(epuck->angle_ball, gain_angle);

/* if((abs(lin_speed) + abs(angle_speed)) > 1000)
{
lin_speed = lin_speed * 999/(abs(lin_speed) + abs(angle_speed));
angle_speed = angle_speed * 999/(abs(lin_speed) + abs(angle_speed));
}
*/
epuck->lin_speed = lin_speed;
epuck->angle_speed = angle_speed;
e_set_speed(lin_speed, angle_speed);
}

//fonction qui calcule la vitesse linaire optimiale  l'aide d'un rgulateur PI
int calc_lin_speed(int distance, int gain)
{
int consigne = 50;
float h = 0.1;
int ti = 3;
int ecart = consigne-distance;
int lin_speed;

ui_lin = ui_lin + h * ecart / ti;
lin_speed = (ecart + ui_lin) * gain;

if(lin_speed >= 1000)
{
ui_lin = 999/gain - ecart;
if(ui_lin > 60) // valeur aberrante vue sur matlab, donc on restreint  40 la valeur de ui
ui_lin = 60.0;
lin_speed = 999;
}
else if(lin_speed <= -1000)
{
ui_lin = -999/gain + ecart;
if(ui_lin < -10) // valeur aberrante vue sur matlab, donc on restreint  -10 la valeur de ui
ui_lin = -10.0;
lin_speed = -999;
}
return lin_speed;
}

//fonction qui calcule la vitesse angulaire optimiale  l'aide d'un rgulateur P
int calc_angle_speed(int pos_pic, int gain)
```

```
{
int consigne = 0;
int angle_speed = 0;
int ecart = consigne - pos_pic;

angle_speed = ecart*gain;

if(angle_speed >= 1000)
angle_speed = 999;
else if(angle_speed <= -1000)
angle_speed = -999;

return angle_speed;
}

//vide le terme intgrateur.
void ARW()
{
ui_lin = 0.0;
}
```

1.4.15 utility.{h,c}

```
void wait(long num);
int getselector();
```

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

void wait(long num) {
long i;
// for(i=0;i<num;i++);
}

int getselector() {
return SELECTOR0 + 2*SELECTOR1 + 4*SELECTOR2 + 8*SELECTOR3;
}
```

2 Library

2.1 a_d

2.1.1 e_accelerometer.{h,c}

```
/*****************************************************************************

Accelerometer sensor of e-puck
Version 1.0 november 2005


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief Accessing the accelerometer sensor data.
 *
 * A little exemple to read the accelerator.
 * \code
 * #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"); }
 *  }
```

```
 * }
 * \endcode
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

#ifndef _ACCELEROMETER
#define _ACCELEROMETER


/* functions */

void e_init_acc(void);   // to be called before starting using accelerometer
void e_get_acc(int *x, int *y, int *z); // to get analog value of accelerometer
#endif
```

```
/***************************************************************************

Accelerometer sensor of e-puck
Version 1.0 november 2005


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief Accessing the accelerometer sensor data.
 *
 * A little exemple to read the accelerator.
 * \code
 * #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"); }
 *  }
 * }
 * \endcode
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

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

/*! \brief Init the accelerometer A/D converter
 * \warning Must be called before starting using accelerometer
 */
void e_init_acc(void)
{
e_init_ad(); // init AD converter module
}

/*! \brief To get the analogic x, y, z axis accelerations
 * \param x A pointer to store the analogic x acceleration
 * \param y A pointer to store the analogic y acceleration
 * \param z A pointer to store the analogic z acceleration
 */
void e_get_acc(int *x, int *y,int *z)
{
T1CONbits.TON = 0;//cut of proximity timer
*x=e_read_ad(ACCX);
*y=e_read_ad(ACCY);
*z=e_read_ad(ACCZ);
T1CONbits.TON = 1;
}
```

2.1.2 e_ad_conv.{h,c}

```
/***************************************************************************
```

```
Analogic/Digital conversion
December 2004: first version Lucas Meier & Francesco Mondada
Version 1.0 november 2005 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2006 Lucas Meier & Francesco Mondada
(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

********************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief Module for the Analogic/Digital conversion.
 * \author Code: Lucas Meier & Francesco Mondada, Michael Bonami \n Doc: Jonathan Besuchet
 */

/*! \defgroup a_d Analogic/Digital conversion (ADC)
 *
 * \section intro_sec Introduction
 * The microcontroller p30f6014A has a 12-bit Analog-to-Digital
 * Converter (ADC). This package is built to manage this ADC.
 * \n The e-puck has three peripherals which take advantage from it.
 * - 1 3D accelerometer
 * - 8 proximity sensors
 * - 3 microphones
 *
 * \section organisation_sec 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 \b arrays containing all the data
 * of all the peripherals which use the ADC (\ref e_mic_scan for the microphones, \ref e_acc_scan
 * for the accelerometer, \ref e_ambient_ir and \ref e_ambient_and_reflected_ir 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,
 * advance_ad_scan/e_prox.c, advance_ad_scan/e_micro.c) are also more elaborates than
 * the same one in the standard package.
 *
 * \author Doc: Jonathan Besuchet
 */

#ifndef _AD_CONV
#define _AD_CONV

/* functions */
void e_init_ad(void); // to be used at the beginning to initialize AD
int e_read_ad(unsigned int channel); // simple function to sample one channel

#endif
```

```
/********************************************************************************

Analogic/Digital conversion
December 2004: first version Lucas Meier & Francesco Mondada
Version 1.0 november 2005 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2006 Lucas Meier & Francesco Mondada
(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

********************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief Module for the Analogic/Digital conversion.
 * \author Code: Lucas Meier & Francesco Mondada, Michael Bonami \n Doc: Jonathan Besuchet
 */


//#include "p30f6014a.h"
#include "../motor_led/e_epuck_ports.h"

/*! \brief Initialize all the A/D register needed */
void e_init_ad(void)
{
  ADCON1 = ADCON2 = ADCON3 = 0;
  // ADPCFGbits.PCFGx
  // = 0 for Analog input mode,
  // = 1 for digital input mode (default)
  ADPCFGbits.PCFG0 = 1; // Debugger
  ADPCFGbits.PCFG1 = 1; // Debugger
  ADPCFGbits.PCFG2 = 0; // mic1
  ADPCFGbits.PCFG3 = 0; // mic2
  ADPCFGbits.PCFG4 = 0; // mic3
  ADPCFGbits.PCFG5 = 0; // axe x acc.
  ADPCFGbits.PCFG6 = 0; // axe y acc.
```

```
  ADPCFGbits.PCFG7 = 0; // axe z acc.
  ADPCFGbits.PCFG8 = 0; // ir0
  ADPCFGbits.PCFG9 = 0; // ir1
  ADPCFGbits.PCFG10 = 0;  // ir2
  ADPCFGbits.PCFG11 = 0;  // ir3
  ADPCFGbits.PCFG12 = 0;  // ir4
  ADPCFGbits.PCFG13 = 0;  // ir5
  ADPCFGbits.PCFG14 = 0;  // ir6
  ADPCFGbits.PCFG15 = 0;  // ir7
  ADCON3 = (2*667/TCY_PIC)-1; //ADCS sampling time Tad minimum 667ns
  ADCHS =0x0007;
  ADCON1bits.ADON = 1;
}

/*! \brief Function to sample an AD channel
 * \param channel The A/D channel you want to sample
 *                Must be between 0 to 15
 * \return The sampled value on the specified channel
 */
int e_read_ad(unsigned int channel)
{
  int delay;
  if(channel > 0x000F) return(0);
  ADCHS = channel;
  ADCON1bits.SAMP = 1;
  for(delay = 0; delay < 40; delay++);
  IFS0bits.ADIF = 0;
  ADCON1bits.SAMP = 0;
  while(!IFS0bits.ADIF);
  return(ADCBUF0);
}
```

2.1.3 e_micro.{h,c}

```
/*****************************************************************************

Microphone sensor of e-puck
Version 1.0 January 2006


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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.
 * \code
 * #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"); }
 *  }
 * }
 * \endcode
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

#ifndef _MICROPHONE
#define _MICROPHONE


/* functions */

void e_init_micro(void);   // to be called before starting using microphone
void e_get_micro(int *m1, int *m2,int *m3); // to get analog value of microphone
#endif
```

```
/*****************************************************************************

Microphone sensor of e-puck
Version 1.0 January 2006


This file is part of the e-puck library license.
```

```
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief Accessing the microphone data.
 *
 * A little exemple which takes the volume of microl and if the
 * sound level is more than 2000. The LED1 is turned on.
 * \code
 * #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"); }
 *  }
 * }
 * \endcode
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

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

/*! \brief Init the microphone A/D converter
 * \warning Must be called before starting using microphone
 */
void e_init_micro(void)
{
e_init_ad(); // init AD converter module
}

/*! \brief To get the m0, m1, m2 microphones's values
 * \param m0 A pointer to store the m0 analogic value
 * \param m1 A pointer to store the m1 analogic value
 * \param m2 A pointer to store the m2 analogic value
 */
void e_get_micro(int *m0, int *m1,int *m2)
{
T1CONbits.TON = 0;//cut of proximity timer
*m0=e_read_ad(MIC1);
*m1=e_read_ad(MIC2);
*m2=e_read_ad(MIC3);
T1CONbits.TON = 1;
}
```

2.1.4 e_prox_timer2.c

```
/*****************************************************************************

Control proximity sensor of e-puck with timer 2
December 2004: first version Lucas Meier & Francesco Mondada
Version 1.0 november 2005 Michael Bonani
Version 1.1 January 2006 Xavier Raemy

This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2006 Lucas Meier & Francesco Mondada
(c) 2006-2007 Michael Bonani & Xavier Raemy

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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 \ref e_get_prox(unsigned int sensor_number) function.
 * \warning The module uses the timer2
 * \sa e_prox.c, e_prox.h to use the timer1
 * \author Code: Lucas Meier & Francesco Mondada, Michael Bonani, Xavier Raemy \n Doc: Jonathan Besuchet
 */
```

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

/* internal variables for prox */
static int ambient_ir[8];  // ambient light measurement
static int ambient_and_reflected_ir[8]; // light when led is on
static int reflected_ir[8];  // variation of light


/* internal calls for prox */
void init_tmr2(void)
{
  T2CON = 0;                    //
  T2CONbits.TCKPS = 1;          // prescsaler = 8
  TMR2 = 0;                     // clear timer 2
  PR2 = (350.0*MICROSEC)/8.0;   // first interrupt after 350us with 8 prescaler
  IFS0bits.T2IF = 0;            // clear interrupt flag
  IEC0bits.T2IE = 1;            // set interrupt enable bit
  T2CONbits.TON = 1;            // start Timer2
}

void __attribute__((interrupt, auto_psv, shadow))
_T2Interrupt(void)
{

// read ambient light and switch on leds in a first phase
// wait 350 us to let the phototransistor react
// read reflected light and switch off the leds in a second phase
// wait 3 ms before stating again
// repeat these two steps for the four couples of prox sensors

  static int ir_phase=0; // phase can be 0 (ambient) or 1 (reflected)
  static int ir_number=0; // number goes from 0 to 3 (4 couples of sensors)

  IFS0bits.T2IF = 0;            // clear interrupt flag

  switch (ir_number)
  {
    case 0: // ir sensors 0 and 4
    {
      if (ir_phase == 0)
      {
        PR2 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
ambient_ir[0] = e_read_ad(IR0);
        ambient_ir[4] = e_read_ad(IR4);
        PULSE_IR0 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR2 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[0] = e_read_ad(IR0);
        ambient_and_reflected_ir[4] = e_read_ad(IR4);
reflected_ir[0] = ambient_ir[0] - ambient_and_reflected_ir[0];
reflected_ir[4] = ambient_ir[4] - ambient_and_reflected_ir[4];
        PULSE_IR0 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 1; // next two sensors
      }
      break;
    }
    case 1: // ir sensors 1 and 5
    {
      if (ir_phase == 0)
      {
        PR2 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
        ambient_ir[1] = e_read_ad(IR1);
        ambient_ir[5] = e_read_ad(IR5);
        PULSE_IR1 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR2 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[1] = e_read_ad(IR1);
        ambient_and_reflected_ir[5] = e_read_ad(IR5);
reflected_ir[1] = ambient_ir[1] - ambient_and_reflected_ir[1];
reflected_ir[5] = ambient_ir[5] - ambient_and_reflected_ir[5];
        PULSE_IR1 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 2; // next two sensors
      }
      break;
    }
    case 2: // ir sensors 2 and 6
    {
      if (ir_phase == 0)
      {
        PR2 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
        ambient_ir[2] = e_read_ad(IR2);
        ambient_ir[6] = e_read_ad(IR6);
        PULSE_IR2 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR2 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[2] = e_read_ad(IR2);
        ambient_and_reflected_ir[6] = e_read_ad(IR6);
reflected_ir[2] = ambient_ir[2] - ambient_and_reflected_ir[2];
reflected_ir[6] = ambient_ir[6] - ambient_and_reflected_ir[6];
        PULSE_IR2 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 3; // next sensor
      }
      break;
```

```
    }
    case 3: // ir sensors 3 and 7
    {
      if (ir_phase == 0)
      {
        PR2 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
        ambient_ir[3] = e_read_ad(IR3);
        ambient_ir[7] = e_read_ad(IR7);
        PULSE_IR3 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR2 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[3] = e_read_ad(IR3);
        ambient_and_reflected_ir[7] = e_read_ad(IR7);
reflected_ir[3] = ambient_ir[3] - ambient_and_reflected_ir[3];
reflected_ir[7] = ambient_ir[7] - ambient_and_reflected_ir[7];
        PULSE_IR3 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 0; // next sensor (back to beginning)
      }
      break;
    }
  }

}

/* ----- user calls ----- */

/*! \brief Init the proxymity sensor A/D converter and the timer2
 * \warning Must be called before starting using proximity sensor
 */
void e_init_prox(void)
{
e_init_ad(); // init AD converter module
init_tmr2(); // init timer 1 for ir processing
}

/*! \brief Stop the acquisition (stop timer2) */
void e_stop_prox(void)
{
 T2CONbits.TON = 0;
 PULSE_IR0=PULSE_IR1=PULSE_IR2=PULSE_IR3=0;
}

/*! \brief To get the analogic proxy sensor value of a specific 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.
 * \param sensor_number The proxy sensor's number that you want the value.
 *                      Must be between 0 to 7.
 * \return The analogic value of the specified proxy sensor
 */
int e_get_prox(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return reflected_ir[sensor_number];
}

/*! \brief To get the analogic ambient light value of a specific sensor
 *
 * This function retur the analogic value of the ambient light measurement.
 * \param sensor_number The proxy sensor's number that you want the value.
 *                      Must be between 0 to 7.
 * \return The analogic value of the specified proxy sensor
 */
int e_get_ambient_light(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return ambient_ir[sensor_number];
}
```

2.1.5 e_prox.{h,c}

```
/*****************************************************************************

Accessing proximity sensor of e-puck with timer 1
December 2004: first version Lucas Meier & Francesco Mondada
Version 1.0 november 2005 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2006 Lucas Meier & Francesco Mondada
(c) 2005-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/
```

```
/*! \file
 * \ingroup a_d
 * \brief 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 \ref e_get_prox(unsigned int sensor_number) function.
 *
 * A little exemple which turn the LED0 when an obstacle is detected
 * by the proximity sensor number 0.
 * \code
 * #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"); }
 *  }
 * }
 * \endcode
 * \warning This module uses the timer1
 * \author Code: Lucas Meier & Francesco Mondada, Michael Bonani \n Doc: Jonathan Besuchet
 */

#ifndef _PROX
#define _PROX


/* functions */

void e_init_prox(void);   // to be called before starting using prox
void e_stop_prox(void); //Stop the timer and put pusle to 0
int e_get_prox(unsigned int sensor_number); // to get a prox value
int e_get_ambient_light(unsigned int sensor_number); // to get ambient light value


#endif
```

```
/************************************************************************

Accessing proximity sensor of e-puck with timer 1
December 2004: first version Lucas Meier & Francesco Mondada
Version 1.0 november 2005 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2006 Lucas Meier & Francesco Mondada
(c) 2005-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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 \ref e_get_prox(unsigned int sensor_number) function.
 *
 * A little exemple which turn the LED0 when an obstacle is detected
 * by the proximity sensor number 0.
 * \code
 * #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"); }
 *  }
 * }
 * \endcode
 * \warning This module uses the timer1
 * \author Code: Lucas Meier & Francesco Mondada, Michael Bonani \n Doc: Jonathan Besuchet
 */

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

/* internal variables for prox */
static int ambient_ir[8];  // ambient light measurement
static int ambient_and_reflected_ir[8];  // light when led is on
static int reflected_ir[8];  // variation of light


/* internal calls for prox */
void init_tmr1(void)
{
  T1CON = 0;                    //
  T1CONbits.TCKPS = 1;          // prescsaler = 8
  TMR1 = 0;                     // clear timer 1
  PR1 = (350.0*MICROSEC)/8.0;   // first interrupt after 350us with 8 prescaler
  IFS0bits.T1IF = 0;            // clear interrupt flag
  IEC0bits.T1IE = 1;            // set interrupt enable bit
  T1CONbits.TON = 1;            // start Timer1
}

void __attribute__((interrupt, auto_psv, shadow))
_T1Interrupt(void)
{
// read ambient light and switch on leds in a first phase
// wait 350 us to let the phototransistor react
// read reflected light and switch off the leds in a second phase
// wait 3 ms before stating again
// repeat these two steps for the four couples of prox sensors

  static int ir_phase=0; // phase can be 0 (ambient) or 1 (reflected)
  static int ir_number=0; // number goes from 0 to 3 (4 couples of sensors)

  IFS0bits.T1IF = 0;            // clear interrupt flag

  switch (ir_number)
  {
    case 0: // ir sensors 0 and 4
    {
      if (ir_phase == 0)
      {
        PR1 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
ambient_ir[0] = e_read_ad(IR0);
        ambient_ir[4] = e_read_ad(IR4);
        PULSE_IR0 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR1 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[0] = e_read_ad(IR0);
        ambient_and_reflected_ir[4] = e_read_ad(IR4);
reflected_ir[0] = ambient_ir[0] - ambient_and_reflected_ir[0];
reflected_ir[4] = ambient_ir[4] - ambient_and_reflected_ir[4];
        PULSE_IR0 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 1; // next two sensors
      }
      break;
    }
    case 1: // ir sensors 1 and 5
    {
      if (ir_phase == 0)
      {
        PR1 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
        ambient_ir[1] = e_read_ad(IR1);
        ambient_ir[5] = e_read_ad(IR5);
        PULSE_IR1 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR1 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[1] = e_read_ad(IR1);
        ambient_and_reflected_ir[5] = e_read_ad(IR5);
reflected_ir[1] = ambient_ir[1] - ambient_and_reflected_ir[1];
reflected_ir[5] = ambient_ir[5] - ambient_and_reflected_ir[5];
        PULSE_IR1 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 2; // next two sensors
      }
      break;
    }
    case 2: // ir sensors 2 and 6
    {
      if (ir_phase == 0)
      {
        PR1 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
        ambient_ir[2] = e_read_ad(IR2);
        ambient_ir[6] = e_read_ad(IR6);
        PULSE_IR2 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR1 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[2] = e_read_ad(IR2);
        ambient_and_reflected_ir[6] = e_read_ad(IR6);
```

```
reflected_ir[2] = ambient_ir[2] - ambient_and_reflected_ir[2];
reflected_ir[6] = ambient_ir[6] - ambient_and_reflected_ir[6];
        PULSE_IR2 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 3; // next sensor
      }
      break;
    }
    case 3: // ir sensors 3 and 7
    {
      if (ir_phase == 0)
      {
        PR1 = (350.0*MICROSEC)/8.0; // next interrupt in 350 us
        ambient_ir[3] = e_read_ad(IR3);
        ambient_ir[7] = e_read_ad(IR7);
        PULSE_IR3 = 1; // led on for next measurement
        ir_phase = 1; // next phase
      }
      else
      {
        PR1 = (2100.0*MICROSEC)/8.0; // next interrupt in 3 ms
        ambient_and_reflected_ir[3] = e_read_ad(IR3);
        ambient_and_reflected_ir[7] = e_read_ad(IR7);
reflected_ir[3] = ambient_ir[3] - ambient_and_reflected_ir[3];
reflected_ir[7] = ambient_ir[7] - ambient_and_reflected_ir[7];
        PULSE_IR3 = 0; // led off
        ir_phase = 0; // reset phase
        ir_number = 0; // next sensor (back to beginning)
      }
      break;
    }
  }

}

/* ---- user calls ---- */

/*! \brief Init the proxymity sensor A/D converter and the timer1
 * \warning Must be called before starting using proximity sensor
 */
void e_init_prox(void)
{
e_init_ad(); // init AD converter module
init_tmr1(); // init timer 1 for ir processing
}

/*! \brief Stop the acquisition (stop timer1) */
void e_stop_prox(void)
{
 T1CONbits.TON = 0;
 PULSE_IR0=PULSE_IR1=PULSE_IR2=PULSE_IR3=0;
}

/*! \brief To get the analogic proxy sensor value of a specific 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.
 * \param sensor_number The proxy sensor's number that you want the value.
 *                      Must be between 0 to 7.
 * \return The analogic value of the specified proxy sensor
 */
int e_get_prox(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return reflected_ir[sensor_number];
}

/*! \brief To get the analogic ambient light value of a specific sensor
 *
 * This function retur the analogic value of the ambient light measurement.
 * \param sensor_number The proxy sensor's number that you want the value.
 *                      Must be between 0 to 7.
 * \return The analogic value of the specified proxy sensor
 */
int e_get_ambient_light(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return ambient_ir[sensor_number];
}
```

2.2 a_d/advance_ad_scan

2.2.1 e_acc.{h,c}

```
/*****************************************************************************

Accessing the accelerometer data (advance)
Novembre 7 2005 Borter Jean-Joel


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45
```

```
(c) 2005 Borter Jean-Joel

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

**********************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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.
 * \n \n Two structures are used:
 * - TypeAccSpheric to store the acceleration data on sherical coordinates.
 * - TypeAccRaw to store the acceleration data on cartesian coordinates.
 *
 * A little exemple to read the accelerator.
 * \code
 * #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"); }
 * }
 *  }
 * \endcode
 * \author Code: Borter Jean-Jol \n Doc: Jonathan Besuchet
 */

#ifndef _ACC_DEFS
#define _ACC_DEFS


#define CST_RADIAN (180.0/3.1415) // used to convert radian in degrees
#define ANGLE_ERROR 666.0 // value returned if an angle can't be defined
#define FILTER_SIZE 5 // define the size of the averaging filter

// ID of the different captor in their respective array

#define ACCX_BUFFER 0
#define ACCY_BUFFER 1
#define ACCZ_BUFFER 2


/*! \struct TypeAccSpheric
 * \brief struct to store the acceleration vector in spherical coord
 */
typedef struct
{
float acceleration; /*!< lenght of the acceleration vector
 * = intensity of the acceleration */
float orientation; /*!< orientation of the acceleration vector
 * in the horizontal plan, zero facing front
 * - 0 = inclination to the front
 * (front part lower than rear part)
 * - 90 = inclination to the left
 * (left part lower than right part)
 * - 180 = inclination to the rear
 * (rear part lower than front part)
 * - 270 = inclination to the right
 * (right part lower than left part) */
float inclination; /*!< inclination angle with the horizontal plan
 * - 0 = e-puck horizontal
 * - 90 = e-puck vertical
 * - 180 = e-puck horizontal but up-side-down */
} TypeAccSpheric;

/*! \struct TypeAccRaw
 * \brief struct to store the acceleration raw data
 * in carthesian coord
 */
typedef struct
{
int acc_x; /*!< The acceleration on x axis */
int acc_y; /*!< The acceleration on y axis */
int acc_z; /*!< The acceleration on z axis */
} TypeAccRaw;


/*****************************************************************
 * ---------------------- Functions from acc.c ------------------------
```

```
 *****************************************************************/

int e_get_acc(unsigned int captor);
int e_get_acc_filtered(unsigned int captor, unsigned int filter_size);
TypeAccSpheric e_read_acc_spheric(void);
float e_read_orientation(void);
float e_read_inclination(void);
float e_read_acc(void);

TypeAccRaw e_read_acc_xyz(void);
int e_read_acc_x(void);
int e_read_acc_y(void);
int e_read_acc_z(void);

void e_acc_calibr(void);
void e_display_angle(void);
#endif
```

```
/*****************************************************************************

Accessing the accelerometer data (advance)
Novembre 7 2005 Borter Jean-Joel


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005 Borter Jean-Joel

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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.
 * \n \n Two structures are used:
 * - TypeAccSpheric to store the acceleration data on sherical coordinates.
 * - TypeAccRaw to store the acceleration data on cartesian coordinates.
 *
 * A little exemple to read the accelerator.
 * \code
 * #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"); }
 * }
 *  }
 * \endcode
 * \author Code: Borter Jean-Joel \n Doc: Jonathan Besuchet
 */

#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>

extern int e_acc_scan[3][ACC_SAMP_NB];
extern unsigned int e_last_acc_scan_id;

/**************************************************
 * internal variables                             *
 **************************************************/
static int angle_mem = 0; //used in the display_angle function

static int centre_x = 0; //zero value for x axe
static int centre_y = 0; //zero value for y axe
static int centre_z = 2000; //zero value for z axe

static int acc_x, acc_y, acc_z; //raw acceleration values in carthesian coord.
static float acceleration, orientation, inclination; //spherical coord. of the accel vector
```

```
/*****************************************************
 * internal function                                 *
 *****************************************************/

/*! \brief Read the last value of a given accelerator axes
 * \param captor ID of the AD channel to read
 * (must be 0 = x, 1 = y or 2 = z)
 * \return value filtered channel's value
 */
int e_get_acc(unsigned int captor)
{
switch(captor) {
case 0: return (e_acc_scan[captor][e_last_acc_scan_id] - centre_x);
case 1: return (e_acc_scan[captor][e_last_acc_scan_id] - centre_y);
case 2: return (e_acc_scan[captor][e_last_acc_scan_id] - centre_z);
default: return((int)ANGLE_ERROR);
}
}

/*! \brief Read the value of a channel, filtered by an averaging filter
 * \param captor ID of the AD channel to read (must be 0 to 2)
 * \param filter_size size of the filter (must be between 1 and SAMPLE_NUMBER)
 * \return value filtered channel's value
 */
int e_get_acc_filtered(unsigned int captor, unsigned int filter_size)
{
long temp = 0;
int i,j;

// channel ID must be between 0 to 13 and
// filter_size must be between 1 to SAMPLE_NUMBER
if ((captor < 3) &&
(filter_size > 0) && (filter_size <= ACC_SAMP_NB))
{
for (i=0, j=(ACC_SAMP_NB-(filter_size-(e_last_acc_scan_id+1)))%ACC_SAMP_NB ; i<filter_size ; i++, j=(j+1)%ACC_SAMP_NB)
{
temp += e_acc_scan[captor][j];
}
}
return ((int)(temp/filter_size));
}

/*! \brief read the x, y, z values, apply an averaging filter
 * and finaly subbstract the center values.
 */
void calculate_acc_raw(void)
{
acc_x = e_get_acc_filtered(ACCX_BUFFER, FILTER_SIZE) - centre_x; // generates positive
acc_y = e_get_acc_filtered(ACCY_BUFFER, FILTER_SIZE) - centre_y; // and negative value
acc_z = e_get_acc_filtered(ACCZ_BUFFER, FILTER_SIZE) - centre_z; // to make the use easy
return;
}

/*! \brief calculate the intensity of the acceleration vector,
 * and the Euler's angles
 */
void calculate_acc_spherical(void)
{
calculate_acc_raw();

// calculat the absolute acceleration value
acceleration = sqrtf((float)(((long)acc_x * (long)acc_x) + ((long)acc_y * (long)acc_y) + ((long)acc_z * (long)acc_z)));

inclination =  90.0 - atan2f((float)(acc_z), sqrtf( (float)(((long)acc_x * (long)acc_x) + ((long)acc_y * (long)acc_y) )) * CST_RADIAN;

if (inclination<5)
orientation=0;
else
orientation = (atan2f((float)(acc_x), (float)(acc_y)) * CST_RADIAN) + 180.0; // 180 is added to have 0 to 360 range

return;
}


/*****************************************************
 * user called function                              *
 *****************************************************/

/*! \brief 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
 */
void e_acc_calibr(void)
{
centre_x = e_get_acc_filtered(ACCX_BUFFER, 50);
centre_y = e_get_acc_filtered(ACCY_BUFFER, 50);
centre_z = e_get_acc_filtered(ACCZ_BUFFER, 50);
}

/*! \brief calculate and return the accel. in spherical coord
 * \return acceleration in spherical coord
 * \sa TypeAccSpheric
 */
TypeAccSpheric e_read_acc_spheric(void)
{
TypeAccSpheric result;

calculate_acc_spherical();
result.acceleration = acceleration;
result.orientation  = orientation;
result.inclination  = inclination;

return(result);
}
```

```
/*! \brief calculate and return the inclination angle
 * \return inclination angle of the robot
 */
float e_read_inclination(void)
{
calculate_acc_spherical();

return(inclination);
}

/*! \brief calculate and return the orientation angle
 * \return orientation of the accel vector
 */
float e_read_orientation(void)
{
calculate_acc_spherical();

return(orientation);
}

/*! \brief calculate and return the intensity of the acceleration
 * \return intensity of the acceleration vector
 */
float e_read_acc(void)
{
calculate_acc_spherical(); //calculate intesnsity and euler's angle

return(acceleration);
}

/*! \brief calculate and return acceleration on the x,y,z axis
 * \return acceleration on the x,y,z axis
 * \sa TypeAccRaw
 */
TypeAccRaw e_read_acc_xyz(void)
{
TypeAccRaw result;

calculate_acc_raw();
result.acc_x = acc_x;
result.acc_y = acc_y;
result.acc_z = acc_z;

return(result);
}

/*! \brief calculate and return acceleration on the x axis
 * \return acceleration on the x axis
 */
int e_read_acc_x(void)
{
calculate_acc_raw();

return(acc_x);
}

/*! \brief calculate and return acceleration on the y axis
 * \return acceleration on the y axis
 */
int e_read_acc_y(void)
{
calculate_acc_raw();

return(acc_y);
}

/*! \brief calculate and return acceleration on the z axis
 * \return acceleration on the z axis
 */
int e_read_acc_z(void)
{
calculate_acc_raw();

return(acc_z);
}

/*! \brief light the led according to the orientation angle */
void e_display_angle(void)
{
float angle = 0.0;

//  To avoid oscillation the limite of variation is limited at
//  a fix value wich is 1/9 of the LED resolution
angle = e_read_orientation();
if ( abs(angle_mem - angle) > 5.0)
{
LED0=LED1=LED2=LED3=LED4=LED5=LED6=LED7=0;
// table of selection
if (    (angle > (360.0 - 22.5)) |  (angle <= (0.0   + 22.5)) && angle != ANGLE_ERROR)
LED0 = 1;
else if ( angle > (45.0 - 22.5)  && angle <= (45.0  + 22.5))
LED7 = 1;
else if ( angle > (90.0 - 22.5)  && angle <= (90.0  + 22.5))
LED6 = 1;
else if ( angle > (135.0 - 22.5) && angle <= (135.0 + 22.5))
LED5 = 1;
else if ( angle > (180.0 - 22.5) && angle <= (180.0 + 22.5))
LED4 = 1;
else if ( angle > (225.0 - 22.5) && angle <= (225.0 + 22.5))
LED3 = 1;
else if ( angle > (270.0 - 22.5) && angle <= (270.0 + 22.5))
LED2 = 1;
else if ( angle > (315.0 - 22.5) && angle <= (315.0 + 22.5))
LED1 = 1;
angle_mem = angle; // value to compare with the next one
}
```

```
}
```

2.2.2 e_ad_conv.{h,c}

```
/***************************************************************************

Advance Analogic/Digital conversion
december 2005: first version Francesco Mondada
april 2006: debug and optimisation Michael Bonani
Borter Jean-Joel


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005 Francesco Mondada
(c) 2006-2007 Michael-Bonani & Borter Jean-Joel

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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.
 * \n \n The data sampled are stored in the corresponding array:
 * - \ref e_mic_scan[3][MIC_SAMP_NB]       Array to store the mic values
 * - \ref e_acc_scan[3][ACC_SAMP_NB]       Array to store the acc values
 * - \ref e_ambient_ir[8]                  Array to store ambient light measurement
 * - \ref e_ambient_and_reflected_ir[8]    Array to store ambient and reflected light measurement
 *
 * In all the files of this module (e_acc.c, 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): \n
 * \code
 * 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);
 * }
 * \endcode
 * \author Code: Francesco Mondada, Michael-Bonani & Borter Jean-Joel \n Doc: Jonathan Besuchet
 */

#ifndef _AD_CONV
#define _AD_CONV

//////////////////////////////////////////////////////////////
// WARNING:  This file is to be used with dsPIC30F6014A    //
//                      with 8x PLL                        //
//////////////////////////////////////////////////////////////

// sampling frequency for the microphones
#define MIC_SAMP_FREQ 16384.0 // WARNING: must be the highest one. This is the base to calculate ad_cycle
//16384.0 is max could also use 12288


// sampling frequency for the accelerometres and proximetres
#define ACC_PROX_SAMP_FREQ 256.0 // WARNING: should be a fraction of MIC_SAMP_FREQ
// to ensure a good timing precision
// lenght of the IR pulse in seconds
#define PULSE_LENGHT 0.0003

//Calculation of the periodes in ad_cycle
//ad_cycle = 1/MIC_SAMP_FREQ
#define ACC_PROX_PERIOD (int)(MIC_SAMP_FREQ/ACC_PROX_SAMP_FREQ) // 60 acc and prox periode in [ad_cycle]
#define PULSE_PERIOD (int)(PULSE_LENGHT*MIC_SAMP_FREQ) //  pulse length in [ad_cycle]

//ADCS calculation to allways have the same time for an AD conversion scan
//with a different numbers of channels
#define ADCS_3_CHAN (int)(2.0*FCY/(MIC_SAMP_FREQ*(14+1)*3)-1) // SAMPLETIME 3TAD
#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 // number of microphone samples to store (put 100 to use with sercom_adv)
#define ACC_SAMP_NB  50 // number of accelerometer samples to store

#define MICRO_ONLY 1
#define ALL_ADC 0

void e_init_ad_scan(unsigned char only_micro);
unsigned char e_ad_is_array_filled(void);
unsigned char e_ad_is_acquisition_completed(void);
void e_ad_scan_on(void);
void e_ad_scan_off(void);

#endif /*_AD_CONV*/

/* End of File : ad_conv.h */
```

```
/***************************************************************************
```

```
Advance Analogic/Digital conversion
december 2005: first version Francesco Mondada
april 2006: debug and optimisation Michael Bonani
Borter Jean-Joel


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005 Francesco Mondada
(c) 2006-2007 Michael-Bonani & Borter Jean-Joel

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*******************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief Module for the advance Analogic/Digital conversion.
 * \author Code: Francesco Mondada, Michael-Bonani & Borter Jean-Joel \n Doc: Jonathan Besuchet
 */

#include "./../../motor_led/e_epuck_ports.h"
#include "e_ad_conv.h"
//#include "./../../fft/e_fft.h"

int e_mic_scan[3][MIC_SAMP_NB]; /*!< Array to store the mic values */
int e_acc_scan[3][ACC_SAMP_NB]; /*!< Array to store the acc values */
unsigned int e_last_mic_scan_id = 0; //ID of the last scan in the mic array (must be int else probleme of overchange)
unsigned int e_last_acc_scan_id = 0; //ID of the last scan in the acc array

int e_ambient_ir[8]; /*!< Array to store the ambient light measurement */
int e_ambient_and_reflected_ir[8]; /*!< Array to store the light when IR led is on */

static unsigned char is_ad_acquisition_completed = 0;
static unsigned char is_ad_array_filled = 0;
static unsigned char micro_only = 0;


/*! \brief 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.
 * \param only_micro Put MICRO_ONLY to use only the three microphones
 * at 33kHz. Put ALL_ADC to use all the stuff using the ADC.
 */
void e_init_ad_scan(unsigned char only_micro)
{
if(only_micro == MICRO_ONLY)
micro_only = MICRO_ONLY;
else
micro_only = ALL_ADC;

ADCON1 = 0; //reset to default value
ADCON2 = 0; //reset to default value
ADCON3 = 0; //reset to default value
ADCHS = 0; //reset to default value

// ADPCFGbits.PCFGx
// = 0 for Analog input mode,
// = 1 for digital input mode (default)
ADPCFGbits.PCFG0 = 1;   // Debugger
ADPCFGbits.PCFG1 = 1;   // Debugger
ADPCFGbits.PCFG2 = 0;   // micro 0
ADPCFGbits.PCFG3 = 0;   // micro 1
ADPCFGbits.PCFG4 = 0;   // micro 2
ADPCFGbits.PCFG5 = 0;   // axe x acc
ADPCFGbits.PCFG6 = 0;   // axe y acc
ADPCFGbits.PCFG7 = 0;   // axe z acc
ADPCFGbits.PCFG8 = 0;   // ir0
ADPCFGbits.PCFG9 = 0;   // ir1
ADPCFGbits.PCFG10 = 0;  // ir2
ADPCFGbits.PCFG11 = 0;  // ir3
ADPCFGbits.PCFG12 = 0;  // ir4
ADPCFGbits.PCFG13 = 0;  // ir5
ADPCFGbits.PCFG14 = 0;  // ir6
ADPCFGbits.PCFG15 = 0;  // ir7

//specifie the channels to be scanned
ADCSSLbits.CSSL0 = 0;   // Debugger
ADCSSLbits.CSSL1 = 0;   // Debugger
ADCSSLbits.CSSL2 = 1;   // micro 0
ADCSSLbits.CSSL3 = 1;   // micro 1
ADCSSLbits.CSSL4 = 1;   // micro 2
ADCSSLbits.CSSL5 = 0;   // axe x acc
ADCSSLbits.CSSL6 = 0;   // axe y acc
ADCSSLbits.CSSL7 = 0;   // axe z acc
ADCSSLbits.CSSL8 = 0;   // ir0
ADCSSLbits.CSSL9 = 0;   // ir1
ADCSSLbits.CSSL10 = 0;  // ir2
ADCSSLbits.CSSL11 = 0;  // ir3
ADCSSLbits.CSSL12 = 0;  // ir4
ADCSSLbits.CSSL13 = 0;  // ir5
ADCSSLbits.CSSL14 = 0;  // ir6
ADCSSLbits.CSSL15 = 0;  // ir7

ADCON1bits.FORM = 0; //output = unsigned int
ADCON1bits.ASAM = 1; //automatic sampling on
ADCON1bits.SSRC = 7; //automatic convertion mode

ADCON2bits.SMPI = 3-1; //interupt on 14th sample
```

```
ADCON2bits.CSCNA = 1; //scan channel input mode on

ADCON3bits.SAMC = 1; //number of cycle between acquisition and conversion (need 2 for the prox)
if(micro_only)
ADCON3bits.ADCS = 19; //acquisition only for micro = 33kHz
else
ADCON3bits.ADCS = ADCS_3_CHAN; //Tad = (ADCS + SAMC) * Tcy/2 = 2170[ns],
//WARNING: Tad min must be 667 [ns]

IFS0bits.ADIF = 0; //Clear the A/D interrupt flag bit
IEC0bits.ADIE = 1; //Set the A/D interrupt enable bit

ADCON1bits.ADON = 1; //enable AD conversion

//wait for the array to be filled one time
if(!micro_only)
do
{
NOP();
}
while (e_last_acc_scan_id < ACC_SAMP_NB-1);
}


/*! \brief Save the AD buffer registers in differents arrays */
void __attribute__((__interrupt__, auto_psv)) _ADCInterrupt(void)
{
is_ad_acquisition_completed = 0;
is_ad_array_filled = 0;

volatile unsigned int * adc_ptr;
static unsigned char period_counter = 0; // period counter for accelerometres and proximetres
static unsigned char prox_number = 0; // number goes from 0 to 3 (4 couples of sensors)

//Clear the A/D Interrupt flag bit or else the CPU will
//keep vectoring back to the ISR
IFS0bits.ADIF = 0;

if(micro_only)
{
//////////////////////////////////////
//  Copy of the buffer regs in the  //
//  approprieted array              //
//////////////////////////////////////
adc_ptr = &ADCBUF0;

e_mic_scan[0][e_last_mic_scan_id] = *adc_ptr++;
e_mic_scan[1][e_last_mic_scan_id] = *adc_ptr++;
e_mic_scan[2][e_last_mic_scan_id] = *adc_ptr++;

if (++e_last_mic_scan_id >= MIC_SAMP_NB) {
e_last_mic_scan_id = 0;
is_ad_array_filled = 1;   // indicate that the array is filled
}
}
else
{
// Normally disable the AD converter in order to be able to modify
//its behevior, but no difference seen. If unable AD, take more time

///////////////////////////////////
// Configure AD reg for next scan //
///////////////////////////////////

// mic channels are always selected for next scan
ADCSSL = 0x001C; // micro selected
ADCON2bits.SMPI = 3-1; //interupt on 3th sample
ADCON3bits.ADCS = ADCS_3_CHAN;

// mic + acc
if (period_counter == ACC_PROX_PERIOD-3) // cycle before last cycle of the periode
{
// acc channels selection
ADCSSL=0x00FC; // mic + acc selected
ADCON2bits.SMPI = 6-1; //interupt on 8th sample
ADCON3bits.ADCS = ADCS_6_CHAN;
}
// mic + prox
else if (period_counter == ACC_PROX_PERIOD-2) // cycle before last cycle of the periode
{
ADCON2bits.SMPI = 5-1; //interupt on 8th sample
ADCON3bits.ADCS = ADCS_5_CHAN;// prox channels selection
switch (prox_number)
{
// ir sensors 0 and 4
case 0: ADCSSL=0x111C;
break;
// ir sensors 1 and 5
case 1: ADCSSL=0x221C;
break;
// ir sensors 2 and 6
case 2: ADCSSL=0x441C;
break;
// ir sensors 3 and 7
case 3: ADCSSL=0x881C;
break;
}
}

// micro + prox
else if (period_counter == PULSE_PERIOD-2) // cycle before last cycle of the periode
{
ADCON2bits.SMPI = 5-1; //interupt on 5th sample
ADCON3bits.ADCS = ADCS_5_CHAN;
switch (prox_number)
{
// ir sensors 0 and 4
```

```
// ir sensors 0 and 4
case 0: ADCSSL=0x111C;
break;
// ir sensors 1 and 5
case 1: ADCSSL=0x221C;
break;
// ir sensors 2 and 6
case 2: ADCSSL=0x441C;
break;
// ir sensors 3 and 7
case 3: ADCSSL=0x881C;
break;
}
}

//reenable the AD converter
//ADCON1bits.ADON = 1;

////////////////////////////////////
//  Copy of the buffer regs in the  //
//        appropriated array        //
////////////////////////////////////
adc_ptr = &ADCBUF0;

//mic channels are always copied
e_mic_scan[0][e_last_mic_scan_id] = *adc_ptr++;
e_mic_scan[1][e_last_mic_scan_id] = *adc_ptr++;
e_mic_scan[2][e_last_mic_scan_id] = *adc_ptr++;

if(e_last_mic_scan_id++>=MIC_SAMP_NB-1)
e_last_mic_scan_id=0;

if (period_counter == ACC_PROX_PERIOD-2)
{
//acc channels copy
e_acc_scan[0][e_last_acc_scan_id] = *adc_ptr++;
e_acc_scan[1][e_last_acc_scan_id] = *adc_ptr++;
e_acc_scan[2][e_last_acc_scan_id] = *adc_ptr;

if(e_last_acc_scan_id++>=ACC_SAMP_NB-1)
e_last_acc_scan_id=0;
}
else if(period_counter == ACC_PROX_PERIOD-1)
{
//prox channels copy (ambient)
switch (prox_number)
{
// prox 0 and 4
case 0: e_ambient_ir[0] = *adc_ptr++;
e_ambient_ir[4] = *adc_ptr;
PULSE_IR0 = 1;
break;
// prox 1 and 5
case 1: e_ambient_ir[1] = *adc_ptr++;
e_ambient_ir[5] = *adc_ptr;
PULSE_IR1 = 1;
break;
// prox 2 and 6
case 2: e_ambient_ir[2] = *adc_ptr++;
e_ambient_ir[6] = *adc_ptr;
PULSE_IR2 = 1;
break;
// prox 3 and 7
case 3: e_ambient_ir[3] = *adc_ptr++;
e_ambient_ir[7] = *adc_ptr;
PULSE_IR3 = 1;
break;
}
}
//prox channels copy (ambient and reflected)
else if (period_counter == PULSE_PERIOD-1)
{
switch (prox_number)
{
// prox 0 and 4
case 0: e_ambient_and_reflected_ir[0] = *adc_ptr++;
e_ambient_and_reflected_ir[4] = *adc_ptr;
PULSE_IR0 = 0;
prox_number = 1;
break;
// prox 1 and 5
case 1: e_ambient_and_reflected_ir[1] = *adc_ptr++;
e_ambient_and_reflected_ir[5] = *adc_ptr;
PULSE_IR1 = 0;
prox_number = 2;
break;
// prox 2 and 6
case 2: e_ambient_and_reflected_ir[2] = *adc_ptr++;
e_ambient_and_reflected_ir[6] = *adc_ptr;
PULSE_IR2 = 0;
prox_number = 3;
break;
// prox 3 and 7
case 3: e_ambient_and_reflected_ir[3] = *adc_ptr++;
e_ambient_and_reflected_ir[7] = *adc_ptr;
PULSE_IR3 = 0;
prox_number = 0;
break;
}
}

if(period_counter++>=ACC_PROX_PERIOD)
period_counter=0;
}
is_ad_acquisition_completed = 1; // indicate a new sample taken
}
```

```
/*! \brief To know if the ADC acquisitionn is completed
 * \return 0 if the new acquisition is not made, 1 if completed.
 */
unsigned char e_ad_is_acquisition_completed(void)
{
return is_ad_acquisition_completed;
}

/*! \brief To know if the ADC acquisitionn of microphone only is completed
 * \return 0 if the new acquisition is not made, 1 if completed.
 */
unsigned char e_ad_is_array_filled(void)
{
return is_ad_array_filled;
}

/*! \brief Enable the ADC conversion
 */
void e_ad_scan_on(void)
{
ADCON1bits.ADON = 1; //enable AD conversion
}

/*! \brief Disable the ADC conversion
 */
void e_ad_scan_off(void)
{
ADCON1bits.ADON = 0; //disable AD conversion
}
```

2.2.3 e_micro.{h,c}

```
/*****************************************************************************

Accessing the microphone data (advance)
Novembre 7 2005 Borter Jean-Joel


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Borter Jean-Joel

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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 \n Doc: Jonathan Besuchet
 */

#ifndef _MICRO
#define _MICRO


// ID of the different captor in their respective array

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

/**************************************************************************
 * -------------------- Functions from micro.c ----------------------
 **************************************************************************/
int e_get_micro(unsigned int micro_id);
int e_get_micro_average(unsigned int micro_id, unsigned int filter_size);
int e_get_micro_volume (unsigned int micro_id);

#endif /*_MICRO*/

/* End of File : ad_conv.h */
```

```
/*****************************************************************************

Accessing the microphone data (advance)
Novembre 7 2005 Borter Jean-Joel


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Borter Jean-Joel

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
```

```
 * \ingroup a_d
 * \brief 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 \n Doc: Jonathan Besuchet
 */

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

/**************************************************
 * external variables                             *
 **************************************************/
extern int e_mic_scan[3][MIC_SAMP_NB];
extern unsigned int e_last_mic_scan_id;


/**************************************************
 * user function                                  *
 **************************************************/

/*! \brief Get the last value of a given micro
 * \param micro_id micro's ID (0, 1, or 2)
 * (use \ref MIC0_BUFFER, \ref MIC1_BUFFER , \ref MIC2_BUFFER  defined in e_micro.h)
 * \return result last value of the micro
 */
int e_get_micro(unsigned int micro_id)
{
return e_mic_scan[micro_id][e_last_mic_scan_id];
}


/*! \brief Get the average on a given number of sample from a micro
 * \param micro_id micro's ID (0, 1, or 2)
 * (use \ref MIC0_BUFFER, \ref MIC1_BUFFER , \ref MIC2_BUFFER  defined in e_micro.h)
 * \param filter_size number of sample to average
 * \return result last value of the micro
 */
int e_get_micro_average(unsigned int micro_id, unsigned int filter_size)
{
long temp = 0;
int i,j;

// channel ID must be between 0 to 2 and
// filter_size must be between 1 to SAMPLE_NUMBER
if ((micro_id < 3) &&
(filter_size > 0) && (filter_size <= MIC_SAMP_NB))
{
for (i=0, j=(MIC_SAMP_NB-(filter_size-(e_last_mic_scan_id+1)))%MIC_SAMP_NB ; i<filter_size ; i++, j=(j+1)%MIC_SAMP_NB)
{
temp += e_mic_scan[micro_id][j];
}
}
return ((int)(temp/filter_size));
}


/*! \brief Get the difference between the highest and lowest sample.
 *
 * \param micro_id micro's ID (0, 1, or 2)
 * (use \ref MIC0_BUFFER, \ref MIC1_BUFFER , \ref MIC2_BUFFER  defined in e_micro.h)
 * \return result volume
 */
int e_get_micro_volume (unsigned int micro_id)
{
int i, highest = 0, lowest = 4999;

for (i=0; i<MIC_SAMP_NB; i++)
{
if (e_mic_scan[micro_id][i] > highest)
{
highest = e_mic_scan[micro_id][i];
}
else if (e_mic_scan[micro_id][i] < lowest)
{
lowest = e_mic_scan[micro_id][i];
}
}
return (highest - lowest);
}


/*! \brief 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.
 * \n [ t ][ t-1 ][ t-2 ][ t-3 ]...[ t-(samples_nb-1) ][ t-samples_nb ]
 * \param micro_id micro's ID (0, 1, or 2)
 * (use \ref MIC0_BUFFER, \ref MIC1_BUFFER , \ref MIC2_BUFFER  defined in e_micro.h)
 * \param *result pointer on the result array
 * \param samples_nb size of the result array
 * (must be between 1 and \ref MIC_SAMP_NB)
 */

void e_get_micro_last_values (int micro_id, int * result, unsigned samples_nb)
{
int i;
if (samples_nb > 0 && samples_nb <= MIC_SAMP_NB)
{
for (i=0 ; i<samples_nb ; i++)
{
result[samples_nb-1 - i] = e_mic_scan[micro_id][(e_last_mic_scan_id + i) % MIC_SAMP_NB];
}
```

```
}
}
```

2.2.4 e_prox.{h,c}

```
/*****************************************************************************

Accessing the proximity sensor data (advance)
Novembre 7 2005 Lucas Meier


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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.
 * \code
 * #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"); }
 *  }
 * }
 * \endcode
 * \author Code: Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef _PROX
#define _PROX

void e_calibrate_ir();
int e_get_prox(unsigned int sensor_number); // to get a prox value
int e_get_calibrated_prox(unsigned int sensor_number);
int e_get_ambient_light(unsigned int sensor_number); // to get ambient light value

#endif
```

```
/*****************************************************************************

Accessing the proximity sensor data (advance)
Novembre 7 2005 Lucas Meier


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup a_d
 * \brief 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 \ref e_get_prox(unsigned int sensor_number) function.
 *
 * A little exemple which turn the LED0 when an obstacle is detected
 * by the proximity sensor number 0.
 * \code
```

```
 * #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"); }
 *  }
 * }
 * \endcode
 * \author Code: Lucas Meier \n Doc: Jonathan Besuchet
 */

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

extern int e_ambient_ir[8]; // ambient light measurement
extern int e_ambient_and_reflected_ir[8]; // light when led is on

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

/*! \brief To calibrate your ir sensor
 * \warning Call this function one time before calling e_get_calibrated_prox
 */
void e_calibrate_ir()
{
int i=0,j=0;
int long t;
int long tmp[8];

for (;i<8;++i) {
init_value_ir[i]=0;
tmp[i]=0;
}

for (;j<100;++j) {
for (i=0;i<8;++i) {
tmp[i]+=(e_get_prox(i));
for (t=0;t<1000;++t);
}
}

for (i=0;i<8;++i) {
init_value_ir[i]=(int)(tmp[i]/(j*1.0));
}
}

/*! \brief 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.
 * \param sensor_number The proxy sensor's number that you want the value.
 *                      Must be between 0 to 7.
 * \return The analogic value of the specified proxy sensor
 */
int e_get_prox(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return e_ambient_ir[sensor_number] - e_ambient_and_reflected_ir[sensor_number];
}

/*! \brief 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().
 * \param sensor_number The proxy sensor's number that you want the calibrated value.
 *                      Must be between 0 to 7.
 * \return The analogic value of the specified proxy sensor
 */
int e_get_calibrated_prox(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return (e_ambient_ir[sensor_number] - e_ambient_and_reflected_ir[sensor_number])
- init_value_ir[sensor_number];
}

/*! \brief To get the analogic ambient light value of a specific ir sensor
 *
 * This function return the analogic value of the ambient light measurement.
 * \param sensor_number The proxy sensor's number that you want the value.
 *                      Must be between 0 to 7.
```

```
 * \return The analogic value of the specified proxy sensor
 */
int e_get_ambient_light(unsigned int sensor_number)
{
if (sensor_number > 7)
return 0;
else
return e_ambient_ir[sensor_number];
}
```

2.3 bluetooth

2.3.1 e_bluetooth.{h,c}

```
/*****************************************************************************

Bluetooth for e-puck
Version 1.0 July 2006 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup bluetooth
 * \brief Manage Bluetooth.
 *
 * This module manage the bluetooth device.
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

/*! \defgroup bluetooth Bluetooth
 *
 * \section intro_sec 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).
 * \n The bluetooth device is connected on the uart1.
 * \n \n 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.
 * \n \n 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 \ref uart module.
 *
 * \author Doc: Jonathan Besuchet
 */

#ifndef _BLUETOOTH
#define _BLUETOOTH

/*! \struct BtDevice
 * \brief general struct for bluetooth device
 */
struct BtDevice
{
  unsigned char address[6]; /*!< Bluetooth MAC address */
  unsigned char class[3]; /*!< Bluetooth class of device*/
  char friendly_name[20]; /*!< The friendly name of the bluetooth device */
};

/*! \struct BtEPuck
 * \brief general struct for other e-puck
 */
struct BtEPuck
{
  unsigned char address[6]; /*!< e-puck's bluetooth MAC address */
  unsigned char number[5]; /*!< e-puck's bluetooth PIN */
};

/*global variable can be access to be read*/
extern unsigned char e_bt_local_paired_device[6*8];
extern 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
* \sa e_bt_find_epuck */
extern 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
* \sa e_bt_find_epuck */

/*------ special e-puck fuction ------*/

int e_bt_find_epuck(void);
char e_bt_connect_epuck(void);


/*------ low level bluettoth fuction ------*/

char e_bt_read_local_pin_number(char *PIN);
char e_bt_read_local_name(char *name);

char e_bt_write_local_pin_number(char *PIN); //give 1 to 16 number PIN (in general e-puck have 4 numbers PIN)
char e_bt_write_local_name(char *name);//write friendly name

char e_bt_etablish_SPP_link(char *address);//established connection to BT address given
char e_bt_release_SPP_link(void);
```

```
char e_bt_send_SPP_data(char *data,char datalenght);//send data maximum 127 bytes, if you are in non transparent mode
char e_bt_tranparent_mode(void);//pass in transparent mode
void e_bt_exit_tranparent_mode(void);//generate break

char e_bt_list_local_paired_device(void);
char e_bt_remove_local_paired_device(int);

int e_bt_inquiry(struct BtDevice *device);//return number of device found
char e_bt_get_friendly_name(struct BtDevice *device);


int e_bt_reset(void);
char  e_bt_factory_reset(void);//WARNING use this function only if you are sure, your e-puck must be than restart, and renamed!!!
#endif
```

```
/*****************************************************************************

Bluetooth for e-puck
Version 1.0 July 2006 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup bluetooth
 * \brief Manage Bluetooth.
 *
 * This module manage the Bluetooth device.
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

/* Bluetooth.c */
#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"

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];


/*------ special e-puck fuction ------*/

/*! \brief Try to find other e-puck
*
* This function make global inguiry and check which device are e-puck,
* and list them in globales tables.
* \return number of e-puck found
* \sa e_bt_present_device, e_bt_present_epuck
*/
int e_bt_find_epuck(void)
{
int device_find, e_puck_find;
int i,j;

e_puck_find=0;
device_find=e_bt_inquiry(e_bt_present_device);

for (i=0;i<device_find;i++)
{
if((e_bt_present_device[i].class[0]==0)&(e_bt_present_device[i].class[1]==0)&(e_bt_present_device[i].class[2]==0)&(e_bt_present_device[i].address[5]==0x08))//marque of e-puck
{
e_bt_get_friendly_name(&e_bt_present_device[i]);
if((e_bt_present_device[i].friendly_name[0]=='e')&(e_bt_present_device[i].friendly_name[1]=='-')&(e_bt_present_device[i].friendly_name[2]=='p'))
{
for(j=0;j<4;j++)
{
e_bt_present_epuck[e_puck_find].address[j]=e_bt_present_device[i].address[j];//copy address
e_bt_present_epuck[e_puck_find].number[j]=e_bt_present_device[i].friendly_name[j+7];//extract number(=pin)
}
e_bt_present_epuck[e_puck_find].number[4]='\0';//end with null
for(j=4;j<6;j++)
e_bt_present_epuck[e_puck_find].address[j]=e_bt_present_device[i].address[j];
e_puck_find++;
}
}
}
return e_puck_find;
}

/* \brief This function will connect to first e-puck found.
 * \return 0 if connect otherwise return error code
 */
char e_bt_connect_epuck(void)
{
int e_puck_find;
char error;
```

```
e_bt_read_local_pin_number(local_bt_PIN);
e_puck_find=e_bt_find_epuck();
if(e_puck_find)
{
e_bt_write_local_pin_number(&e_bt_present_epuck[0].number[0]);
error=e_bt_etablish_SPP_link(&e_bt_present_epuck[0].address[0]);
e_bt_write_local_pin_number(local_bt_PIN);
return error;
}
else
return 99;
}


/*------ low level bluettoth fuction ------*/

/*! \brief Reset the bluetooth module
 * \return The version number
 */
int e_bt_reset(void)
{
char send[7];
char read[30];
int i;
char c;
char version[5];

send[0]=0x02; //send reset request
send[1]=0x52;
send[2]=0x26;
send[3]=0x00;
send[4]=0x00;
send[5]=0x78;
send[6]=0x03;
e_send_uart1_char(send,7);

i=0;
c=0;

 do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';

for(i=0;i<read[6];i++) //extract version
version[i]=read[i+7];
version[i]= '\0';
return atoi(version);
}

/*! \brief Factory reset of the bluetooth module
 * \warning use this function only if you are sure, your e-puck must
 * be restarted, and renamed!!!
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_factory_reset(void)
{
char send[8];
char read[10];
int i;
char c;

send[0]=0x02;
send[1]=0x52;
send[2]=0x1A;
send[3]=0x00;
send[4]=0x00;
send[5]=0x6c;
send[6]=0x03;//link number 1-30
e_send_uart1_char(send,7);

 i=0;
 c=0;
do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
return read[6]; //return error 0=no error
}

/*! \brief Change to transparent mode
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_tranparent_mode(void)
{
char send[8];
char read[10];
int i;
char c;

send[0]=0x02;
send[1]=0x52;
send[2]=0x11;
send[3]=0x01;
send[4]=0x00;
```

```
send[5]=0x64;
send[6]=0x01;//link number 1-30
send[7]=0x03;
e_send_uart1_char(send,8);

 i=0;
 c=0;
do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
return read[6]; //return error 0=no error
}


/*! \brief Exit from the transparent mode */
void e_bt_exit_tranparent_mode(void)
{
long i;
U1MODEbits.UARTEN=0;//disable uart1
_TRISF3=0;
_LATF3=0;//uart 1 TX
for(i=0;i<MILLISEC;i++);//wait at least 1ms
_LATF3=1;//uart 1 TX
_TRISF3=1;
e_init_uart1();
}

/*! \brief Read the PIN number of this e-puck's bluetooth module
 * \param PIN A pointer to store the PIN number
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_read_local_pin_number(char *PIN)
{
char send[7];
unsigned char read[30];
int i;
char c;

send[0]=0x02; //send PIN request
send[1]=0x52;
send[2]=0x16;
send[3]=0x00;
send[4]=0x00;
send[5]=0x68;
send[6]=0x03;
e_send_uart1_char(send,7);

 i=0;
 c=0;

 do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';

for(i=0;i<read[7];i++) //extract PIN
PIN[i]=read[i+8];
PIN[i]= '\0';
return read[6];//return error 0=no error
}

/*! \brief Read the name of this e-puck's bluetooth module
 * \param name A pointer to store the name
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_read_local_name(char *name)
{
char send[7];
char read[40];
int i;
char c;

send[0]=0x02; //send Name request
send[1]=0x52;
send[2]=0x03;
send[3]=0x00;
send[4]=0x00;
send[5]=0x55;
send[6]=0x03;
e_send_uart1_char(send,7);

 i=0;
 c=0;

 do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
```

```
for(i=0;i<read[7];i++) //extract Name
name[i]=read[i+8];

return read[6];//return error 0=no error
}

/*! \brief Write the PIN number on this e-puck's bluetooth module
 * \param PIN A pointer to store the PIN number
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_write_local_pin_number(char *PIN)
{
char send[30];
char read[10];
int i;
char c;
int numberlenght;

numberlenght=strlen(PIN);
//send_uart2(PIN,numberlenght);
send[0]=0x02;
send[1]=0x52;
send[2]=0x17;
send[3]=numberlenght+1;
send[4]=0x00;
send[5]=(send[1]+send[2]+send[3]);
send[6]=numberlenght;
for (i=0;i<numberlenght;i++)
send[i+7]=PIN[i];
send[7+numberlenght]=0x03;
e_send_uart1_char(send,numberlenght+8);

 i=0;
 c=0;

 do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
return read[6];//return error 0=no error
}

/*! \brief Write the name on this e-puck's bluetooth module
 * \param name A pointer to store the name
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_write_local_name(char *name)
{
char send[40];
char read[10];
int i;
char c;
int namelenght;

namelenght=strlen(name);
namelenght++;//add null caracter
//send_uart2(PIN,numberlenght);
send[0]=0x02; //send PIN request
send[1]=0x52;
send[2]=0x04;
send[3]=namelenght+1;
send[4]=0x00;
send[5]=(send[1]+send[2]+send[3]);
send[6]=namelenght;
for (i=0;i<namelenght;i++)
send[i+7]=name[i];
send[7+namelenght]=0x03;
e_send_uart1_char(send,namelenght+8);

 i=0;
 c=0;

 do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
return read[6];//return error 0=no error
}

/*! \brief Research all the accessible bluetooth devices
 * \param device A pointer to BtDevice to store all the caracteristics
 * of each devices found
 * \return the number of device found
 * \sa BtDevice, e_bt_present_device
 */
int e_bt_inquiry(struct BtDevice *device)
{
char send[10];
char read[50];
int devicefound;
int i,j;
char c;

send[0]=0x02; //send Name request
```

```
send[1]=0x52;
send[2]=0x00;
send[3]=0x03;
send[4]=0x00;
send[5]=0x55;
send[6]=0x05;//0x01-0x30 time of inquiri in sec
send[7]=0x0A;//number of maximu response 0=infinite
send[8]=0x00;//mode
send[9]=0x03;
e_send_uart1_char(send,10);
devicefound=0;
 i=j=0;
 c=0;
do{
  i=0;
  do
     {
       if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
     }
    while (((char)c != 0x03)||(i<(read[3]+6)));
     if(read[1]==0x69)
     { for(i=0;i<6;i++) //extract BTaddress
device[devicefound].address[i]=read[i+6];
for(i=0;i<3;i++) //extract BTaddress
device[devicefound].class[i]=read[i+12];
devicefound++;
}
}
    while((read[1]!=0x43)&(read[2]!=0x00));
return devicefound;//return number of device found
}

/*! \brief To get the friendly name of a bluetooth device
 * \param device A pointer on the device that you want the name
 * \return bluetooth error if one occur, 0 otherwise
 * \sa BtDevice
 */
char e_bt_get_friendly_name(struct BtDevice *device)
{
char send[13];
char read[50];

int i;
char c;

send[0]=0x02; //send Name request
send[1]=0x52;
send[2]=0x02;
send[3]=0x06;
send[4]=0x00;
send[5]=0x5a;
send[6]=device[0].address[0];//address of device
send[7]=device[0].address[1];
send[8]=device[0].address[2];
send[9]=device[0].address[3];
send[10]=device[0].address[4];
send[11]=device[0].address[5];
send[12]=0x03;
e_send_uart1_char(send,13);

i=0;
c=0;
do
    {
     if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';

if (read[6]==0)
for(i=0;i<read[13];i++) //extract Name
device[0].friendly_name[i]=read[i+14];
return read[6]; //return error 0=no error
}

/*! \brief Try to connect to another bluetooth device
 * \param address A pointer on the device address which you want
 * to connect
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_etablish_SPP_link(char *address)
{
char send[15];
char read[50];

int i;
char c;

send[0]=0x02; //send Name request
send[1]=0x52;
send[2]=0x0a;
send[3]=0x08;
send[4]=0x00;
send[5]=0x64;
send[6]=0x01;
send[7]=address[0];//address of device
send[8]=address[1];
send[9]=address[2];
send[10]=address[3];
```

```
send[11]=address[4];
send[12]=address[5];
send[13]=0x01;
send[14]=0x03;
e_send_uart1_char(send,15);

i=0;
c=0;
do{
  i=0;
  do
     {
       if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
     }
    while (((char)c != 0x03)||(i<(read[3]+6)));
}
while((read[2]!=0x69)&(read[2]!=0x0B));//spp link etablished

return read[6]; //return rfcomm error 0=no error
}

/*! \brief Unconnect from the current bluetooth device
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_release_SPP_link(void)
{
char send[8];
char read[10];
int i;
char c;

send[0]=0x02;
send[1]=0x52;
send[2]=0x0d;
send[3]=0x01;
send[4]=0x00;
send[5]=0x60;
send[6]=0x01;//link number 1-30
send[7]=0x03;
e_send_uart1_char(send,8);

 i=0;
 c=0;
do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
if((read[2]==0x0d)&(read[6]!=0x00))//control confirm request (in case no link etablished)
return read[6]; //return error 0=no error


do{
  i=0;
  do
     {
       if (e_getchar_uart1(&read[i]))//read response
{
c=read[i];
      i++;
}
     }
    while (((char)c != 0x03)||(i<(read[3]+6)));
}
while((read[1]!=0x69)&(read[2]!=0x051));//spp link released

return read[6]; //return rfcomm error 0=no error
}

/*! \brief Send data to the current bluetooth device
 * \warning send maximum 127 bytes if you are in non transparent mode
 * \param data The datas to send
 * \param datalength The length of the datas to send
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_send_SPP_data(char *data, char datalength)
{
char send[120];
char read[10];
int i;
char c;

//send_uart2(PIN,numberlenght);
send[0]=0x02; //send PIN request
send[1]=0x52;
send[2]=0x0F;
send[3]=datalength+3;
send[4]=0x00;
send[5]=(send[1]+send[2]+send[3]);
send[6]=0x01;//local port
send[7]=datalength;
send[8]=0x00;
for (i=0;i<datalength;i++)
send[i+9]=data[i];
send[9+datalength]=0x03;
e_send_uart1_char(send,datalength+10);

 i=0;
```

```
 c=0;

 do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
return read[6];//return error 0=no error
}

/*! \brief Make a list of the bluetooth address of paired device
 * \return The number of device found
 * \sa e_bt_local_paired_device
 */
char e_bt_list_local_paired_device(void)
{
char send[7];
char read[(8*6)+10];
int devicefound;
int i;
char c;

send[0]=0x02;
send[1]=0x52;
send[2]=0x1c;
send[3]=0x00;
send[4]=0x00;
send[5]=0x6e;
send[6]=0x03;
e_send_uart1_char(send,7);
devicefound=0;
 i=0;
 c=0;
do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
     i++;
}
    }
   while (((char)c != 0x03)||(i<(read[3]+6)));
    devicefound=read[7];
    if(devicefound!=0)
    {
    for(i=0;i<6*devicefound;i++) //extract BTaddress
e_bt_local_paired_device[i]=read[i+8];
}

return devicefound;//return number of device found
}

/*! \brief Remove a paired bluetooth device
 * \param j The number of the paired device to remove from the
 * e_bt_local_paired_device array.
 * \return bluetooth error if one occur, 0 otherwise
 */
char e_bt_remove_local_paired_device(int j)
{
char send[13];
char read[10];
int i;
char c;

send[0]=0x02; //send Name request
send[1]=0x52;
send[2]=0x1b;
send[3]=0x06;
send[4]=0x00;
send[5]=0x73;
send[6]=e_bt_local_paired_device[0+j*6];//address of device
send[7]=e_bt_local_paired_device[1+j*6];
send[8]=e_bt_local_paired_device[2+j*6];
send[9]=e_bt_local_paired_device[3+j*6];
send[10]=e_bt_local_paired_device[4+j*6];
send[11]=e_bt_local_paired_device[5+j*6];
send[12]=0x03;
e_send_uart1_char(send,13);

 i=0;
 c=0;
do
    {
      if (e_getchar_uart1(&read[i])) //read response
{
c=read[i];
      i++;
}
    }
    while (((char)c != 0x03)||(i<(read[3]+6)));
    read[i]='\0';
return read[6]; //return error 0=no error
}
```

2.4 camera/fast_2_timer

2.4.1 e_calc.c

```
/*! \file
 * \ingroup camera1
 * \brief Calculate the timing for the camera (two timers)
 * \author Philippe Rtornaz
 */

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

#define  ARRAY_ORIGINE_X 210
#define ARRAY_ORIGINE_Y 7

#define IRQ_PIX_LAT 1

/*! 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.
 * \param sensor_x1 The X coordinate of the window's corner
 * \param sensor_y1 The Y coordinate of the window's corner
 * \param sensor_width the Width of the interest area, in FULL sampling scale
 * \param sensor_height The Height of the insterest area, in FULL sampling scale
 * \param zoom_fact_width The subsampling to apply for the window's Width
 * \param zoom_fact_height The subsampling to apply for the window's Height
 * \param color_mode The color mode in which the camera should be configured
 * \return Zero if the settings are correct, non-zero if an error occur
 * \sa e_po3030k_write_cam_registers
 */

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 pbp_h, pbp_w;
int nb_pixels, nb_lines;
int real_zoom_h, real_zoom_w;
int sampl_mode;

sensor_x1 += ARRAY_ORIGINE_X;
sensor_y1 += ARRAY_ORIGINE_Y;
/* testing if the mode is acceptable */
if(zoom_fact_height < 1 || zoom_fact_width < 1)
return -1;

/* Check if the area is out of bound */
if((sensor_x1 + sensor_width) > (ARRAY_ORIGINE_X + ARRAY_WIDTH))
return -1;
if((sensor_y1 + sensor_height) > (ARRAY_ORIGINE_Y + ARRAY_HEIGHT))
return -1;

/* Check if Sensor[Width|Height] is a multiple of Zoom */
if(sensor_width % zoom_fact_width)
return -1;
if(sensor_height % zoom_fact_height)
return -1;

/* Search the best subsampling aviable */
if(!(zoom_fact_height%4)) {
if(!(zoom_fact_width%4)) {
sampl_mode = MODE_QQVGA;
real_zoom_h = zoom_fact_height >> 2;
real_zoom_w = zoom_fact_width >> 2;
/* this is done because we must have Vsync a real
 * row before the real picture */
sensor_y1 -= 4;
/* We need Hsync some time before the first effective pixel */
sensor_x1 -= IRQ_PIX_LAT*4;
} else {
if(!(zoom_fact_width%2)) {
sampl_mode = MODE_QVGA;
real_zoom_h = zoom_fact_height >> 1;
real_zoom_w = zoom_fact_width >> 1;
sensor_y1 -= 2;
sensor_x1 -= 2*IRQ_PIX_LAT;
} else {
sampl_mode = MODE_VGA;
real_zoom_h = zoom_fact_height;
real_zoom_w = zoom_fact_width;
sensor_y1--;
sensor_x1 -= IRQ_PIX_LAT;
}
}
} else if(!(zoom_fact_height%2)) {
if(!(zoom_fact_width%2)) {
sampl_mode = MODE_QVGA;
real_zoom_h = zoom_fact_height >> 1;
real_zoom_w = zoom_fact_width >> 1;
sensor_y1 -= 2;
sensor_x1 -= 2*IRQ_PIX_LAT;
} else {
sampl_mode = MODE_VGA;
real_zoom_h = zoom_fact_height;
real_zoom_w = zoom_fact_width;
sensor_y1--;
sensor_x1 -= IRQ_PIX_LAT;
}
} else {
sampl_mode = MODE_VGA;
real_zoom_w = zoom_fact_width;
```

```
real_zoom_h = zoom_fact_height;
sensor_y1--;
sensor_x1 -= IRQ_PIX_LAT;
}
pbp_w = real_zoom_w - 1;
pbp_h = real_zoom_h - 1;

nb_pixels = sensor_width / zoom_fact_width;
nb_lines = sensor_height/ zoom_fact_height;

/* set camera configuration */
if(e_po3030k_set_color_mode(color_mode))
return -1;

if(e_po3030k_set_sampling_mode(sampl_mode))
return -1;

if(color_mode == GREY_SCALE_MODE) {
switch (sampl_mode) {
case MODE_VGA:
e_po3030k_set_speed(SPEED_8);
break;
case MODE_QVGA:
e_po3030k_set_speed(SPEED_4);
break;
case MODE_QQVGA:
e_po3030k_set_speed(SPEED_2);
break;
}
} else {
switch (sampl_mode) {
case MODE_VGA:
e_po3030k_set_speed(SPEED_16);
break;
case MODE_QVGA:
e_po3030k_set_speed(SPEED_8);
break;
case MODE_QQVGA:
e_po3030k_set_speed(SPEED_4);
break;
}
}

if(e_po3030k_set_wx(sensor_x1,ARRAY_ORIGINE_X + ARRAY_WIDTH + 1))
return -1;

if(e_po3030k_set_wy(sensor_y1,ARRAY_ORIGINE_Y + ARRAY_HEIGHT + 1))
return -1;

if(e_po3030k_set_vsync(sensor_y1,ARRAY_ORIGINE_Y + ARRAY_HEIGHT, sensor_x1 + 1))
return -1;

/* set timer configuration */
if(e_po3030k_apply_timer_config(nb_lines,nb_pixels,e_po3030k_get_bytes_per_pixel(color_mode),pbp_w,pbp_h))
return -1;

return 0;

}

/*! Return the number of bytes per pixel in the given color mode
 * \param color_mode The given color mode
 * \return The number of bytes per pixel in the given color mode
 */
int e_po3030k_get_bytes_per_pixel(int color_mode) {
switch (color_mode) {
case GREY_SCALE_MODE:
return 1;
case RGB_565_MODE:
return 2;
case YUV_MODE:
return 2; /* YUV mode is not the best mode for subsampling */
}
/* UNKNOW ! */
return 1;
}

/*! Initialize the camera, must be called before any other function
 */
void e_po3030k_init_cam(void) {
int i;
e_init_port();
e_i2cp_init();
CAM_RESET=0;
for(i=100;i;i--) __asm__ volatile ("nop");
CAM_RESET=1;
for(i=100;i;i--) __asm__ volatile ("nop");
/* enable interrupt nesting */
INTCON1bits.NSTDIS = 0;
/* set a higher priority on camera's interrupts */
IPC5 = (IPC5 & 0xF00F) + 0x0660;
}
```

2.4.2 e_interrupt.s

```
.include "p30f6014A.inc"

; assembler file for the HSYNC interrupt
.global __T4Interrupt
__T4Interrupt:
; /!\ interrupt nesting, push.s allowed only once !
push.s
```

```
bclr  IFS1,#T4IF ; clear interrupt flag
clr  TMR4 ; clear timer

; for fine syncronisation
nop

mov __po3030k_buffer, w1
mov #__po3030k_line_conf, w2
mov #1, w3 ; used for comparaisons

restart_loop:
cp.b w3,[w2++]
bra Z, take_it  ; if w2 == 1
bra LT, end_line ; if w2 == 2
; w2 == 0
nop
nop
nop
bra restart_loop
take_it:
mov PORTD, w0
lsr w0,#8,w0
mov.b w0,[w1++]
bra restart_loop

end_line:
mov w1, __po3030k_buffer
inc __po3030k_current_row
mov __po3030k_current_row,w0
cp __po3030k_row
bra NZ, go_out
; tell others we are ready
mov #1, w0
mov w0, __po3030k_img_ready
; disable ourself
bclr T4CON,#TON
go_out:
pop.s
retfie
```

2.4.3 e_po3030k.h

```
/*! \file
 * \ingroup camera1
 * \brief PO3030k library header (two timers)
 * \author Philippe Rtornaz
 */

/*! \defgroup camera1 Camera fast two timers
 *
 * \section intro_sec 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(), 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().
 * This is typically done after every configuration call and before acquire
 * the first picture.
 *
 * \section defset 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.
 *
 * \section perfsec 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
 *
 * \section IntegrDet 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 !
 *
 * \section example_sect Examples
 *
 * \subsection ex1_sect Basic example
 *
 * \code
#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;
}
\endcode

 * 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.
 *
 * \subsection ex2_sect More advanced example
\code
#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;
}
\endcode
 * 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.
 */

#ifndef __PO3030K_H__
#define __PO3030K_H__

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

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);
```

```
#if PO3030K_FULL

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);

#endif

#endif
```

2.4.4 e_registers.c

```
/*! \file
 * \brief Manage po3030k registers (two timers)
 * \author Philippe Rtornaz
 */



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

#define MCLK ((long) 14745600)  /* 14.7456Mhz */
#define MCLK_P_NS 0.067816840278 /* Master clock period in ns */


/* these two define should go into a private header but, well,
 * I won't do it one for only two define ... */
#define  ARRAY_ORIGINE_X 210
#define ARRAY_ORIGINE_Y 7

/* Some preprocessor ugliness */
#if PO3030K_FULL
#define BASE_D1 0
#define BASE_D2 0
#define BASE_D3 0
#define BASE_D4 0
#else
#define BASE_D1 20
#define BASE_D2 (BASE_D1 + 4)
#define BASE_D3 (BASE_D2 + 8)
#define BASE_D4 (BASE_D3 + 76)
#endif
```

```
#define DEVICE_ID   0xDC

static unsigned char cam_reg[] = {
/* format:
   address,   value */
#define FRAME_WIDTH 0x0353
0x04,(unsigned char) (FRAME_WIDTH >> 8),
0x05,(unsigned char) FRAME_WIDTH,
#define FRAME_HEIGTH 0x01e8
0x06,(unsigned char) (FRAME_WIDTH >> 8),
0x07,(unsigned char) FRAME_WIDTH,
#define WINDOW_X1_BASE 9
0x08,0x00,
0x09,0xd1,
#define WINDOW_Y1_BASE 13
0x0a,0x00,
0x0b,0x07,
#define WINDOW_X2_BASE 17
0x0c,0x03,
0x0d,0x51,
#define WINDOW_Y2_BASE 21
0x0e,0x01,
0x0f,0xe7,

#if PO3030K_FULL
#define BIAS_BASE 25
0x12,0x02,
0x13,0x02,
#define COLGAIN_BASE 29
0x15,0x10,
0x16,0x40,
0x17,0x40,
0x18,0x40,
0x19,0x40,
#define INTEGR_BASE 39
0x1a,0x03,
0x1b,0xe6,
0x1c,0x00,
#endif

#define SPEED_ADDR (45 - BASE_D1)
0x1d,/*0x00*/ 0b01110000,

#if PO3030K_FULL
#define MIRROR_BASE 47
0x1e,0x0a,
0x1f,0x1b,
#endif

#define SAMPLING_ADDR ( 51 - BASE_D2 )
0x20,MODE_VGA,
#define ADCOFF_BASE ( 53 - BASE_D2 )
0x3b,0x00,
#define FLICKM_BASE ( 55 - BASE_D2 )
0x46,0x80,
/* see datasheet p. 30 */
0x48,(unsigned char) (1667./(MCLK_P_NS*256)),

#if PO3030K_FULL
#define FLICKP_BASE 59
0x49,0x96,
0x4a,0x00,
0x4b,0x7d,
0x4c,0x00,
#endif
#define COLOR_M_ADDR (67 - BASE_D3)
#define MODE_R5G6B5 0x08
#define MODE_YUV 0x02
#define MODE_GRAYSCALE 0x0c
0x4e,MODE_R5G6B5,

#if PO3030K_FULL
#define SEPIA_BASE 69
0x4f,0x0a,
0x50,0x70,
#define LENSG_BASE 73
0x59,0x00,
0x5a,0x00,
0x5b,0x00,
#define EDGE_BASE 79
0x5c,0xac,
0x5d,0x02,
#define GAMMA_BASE 83
0x76,0x00,
0x77,0x1a,
0x78,0x2a,
0x79,0x37,
0x7a,0x42,
0x7b,0x56,
0x7c,0x68,
0x7d,0x87,
0x7e,0xa3,
0x7f,0xbc,
0x80,0xd4,
0x81,0xea,
#define COLOR_COEF_BASE 107
0x8e,0x38,
0x8f,0xa5,
0x90,0x0d,
0x91,0x93,
0x92,0x2d,
0x93,0x06,
0x94,0x83,
0x95,0xaa,
0x96,0x4d,
#define CBCRGAIN_BASE 125
0x97,0x25,
```

```
0x98,0x25,
#define BRICTR_BASE 129
0x9b,0x00,
0x9c,0x96,
#define SEPIATONE_BASE 133
0x9f,0x80,
0xa0,0x80,
0xa1,0xb6,
0xa2,0x9d,
0xa3,0xab,
0xa4,0x80,
#endif

#define VSYNCSTART_BASE (145 - BASE_D4)
0xa5,0x00,
0xa6,0x07,
#define VSYNCSTOP_BASE (149 - BASE_D4)
0xa7,0x01,
0xa8,0xe7,
#define VSYNCCOL_BASE (153 - BASE_D4)
0xa9,0x00,
0xaa,0x01,

#if PO3030K_FULL
#define WW_BASE 157
0xad,0x9f,
#define AWVAETOL_BASE 159
0xae,0x22,
#define AESPEED_BASE 161
0xaf,0x8c,
#define EXPOSURE_BASE 163
0xb0,0x07,
0xb1,0xcc,
0xb2,0x00,
#define REFREPO_BASE 169
0xb3,0x70,
0xb9,0x03,
0xba,0xe6,
#define MINMAXEXP_BASE 175
0xbb,0x07,
0xbc,0xcc,
0xbd,0x00,
0xbe,0x0c,
#define MINMAXAWB_BASE 183
0xc1,0x10,
0xc2,0xe0,
0xc3,0x10,
0xc4,0xe0,
0xc5,0x80,
0xc6,0x80,
#define WEIGHWIN_BASE 195
0xc7,0x01,
0xc8,0xa5,
0xc9,0x02,
0xca,0x7a,
0xcb,0x00,
0xcc,0xa7,
0xcd,0x01,
0xce,0x47,
#define AWBAEENABLE_BASE 211
0xd4,0x3c,
0xd5,0x01,
#define GAMMASELCOL_BASE 215
0xd8,0x80
#endif
};
#define NB_REGISTERS (sizeof(cam_reg)/(2*sizeof(cam_reg[0])))


/*! 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.
 * \sa e_po3030k_read_cam_registers
 */
void e_po3030k_write_cam_registers(void) {
int i;
e_i2cp_enable();
for(i=0;i < 2*NB_REGISTERS; i+=2 )
e_i2cp_write(DEVICE_ID,cam_reg[i],cam_reg[i+1]);

e_i2cp_disable();
}

/*! Read the camera register
 * \sa e_po3030k_write_cam_registers
 */
void e_po3030k_read_cam_registers(void) {
int i;
e_i2cp_enable();
for(i=0;i < 2*NB_REGISTERS; i+=2 )
cam_reg[i+1] = e_i2cp_read(DEVICE_ID,cam_reg[i]);
e_i2cp_disable();
}

/*! Set the camera color mode
 * \warning This is an internal function, use e_po3030k_config_cam
 * \param mode The color mode
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p. 31, e_po3030k_write_cam_registers and e_po3030k_config_cam
 */
int e_po3030k_set_color_mode(int mode) {
switch (mode) {
case GREY_SCALE_MODE:
cam_reg[COLOR_M_ADDR] = MODE_GRAYSCALE;
break;
case RGB_565_MODE:
```

```
cam_reg[COLOR_M_ADDR] = MODE_R5G6B5;
break;
case YUV_MODE:
cam_reg[COLOR_M_ADDR] = MODE_YUV;
break;
default:
return -1;
}
return 0;
}

/*! Set the camera sampling mode
 * \warning This is an internal function, use e_po3030k_config_cam
 * \param mode The given sampling mode
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p. 28 and e_po3030k_config_cam
 */
int e_po3030k_set_sampling_mode(int mode) {
if(mode == MODE_VGA || mode == MODE_QVGA ||
mode == MODE_QQVGA )
{
cam_reg[SAMPLING_ADDR] = mode;
return 0;
}
return -1;
}

/*! Set the camera speed
 * \warning This is an internal function, use e_po3030k_config_cam
 * \param mode The given speed
 * \return Zero if OK, non-zero if unknow mode
 * \sa Datasheet p. 26 and e_po3030k_config_cam
 */

int e_po3030k_set_speed(int mode) {
if(mode == SPEED_2 || mode == SPEED_4 ||
mode == SPEED_8 || mode == SPEED_16 ||
mode == SPEED_32 || mode == SPEED_64 ||
mode == SPEED_64 || mode == SPEED_2_3)
{
cam_reg[SPEED_ADDR] = mode;
return 0;
}

return -1;
}
/*! Set the camera window X coordinate
 * \warning This is an internal function, use e_po3030k_ConfigCam
 * \param start The start column
 * \param stop The stop column
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.20-21, e_po3030k_write_cam_registers, e_po3030k_set_wy and e_po3030k_config_cam
 */
int e_po3030k_set_wx(unsigned int start, unsigned int stop) {
unsigned char start_h,start_l,stop_h,stop_l;
if(start >= stop)
return -1;
if(stop > 899)
return -1;
start--;
start_l = (unsigned char) start;
start_h = (unsigned char) (start >> 8);
stop_l = (unsigned char) stop;
stop_h = (unsigned char) (stop >> 8);

cam_reg[WINDOW_X1_BASE] = start_h;
cam_reg[WINDOW_X1_BASE + 2] = start_l;

cam_reg[WINDOW_X2_BASE] = stop_h;
cam_reg[WINDOW_X2_BASE + 2] = stop_l;

return 0;
}

/*! Set the camera window Y coordinate
 * \warning This is an internal function, use e_po3030k_ConfigCam
 * \param start The start row
 * \param stop The stop row
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam
 */
int e_po3030k_set_wy(unsigned int start, unsigned int stop) {
unsigned char start_h,start_l,stop_h,stop_l;
if(start >= stop)
return -1;
if(stop > 499)
return -1;
start_l = (unsigned char) start;
start_h = (unsigned char) (start >> 8);
stop_l = (unsigned char) stop;
stop_h = (unsigned char) (stop >> 8);

cam_reg[WINDOW_Y1_BASE] = start_h;
cam_reg[WINDOW_Y1_BASE + 2] = start_l;

cam_reg[WINDOW_Y2_BASE] = stop_h;
cam_reg[WINDOW_Y2_BASE + 2] = stop_l;

return 0;
}

/*! Set the camera window VSYNC coordinate
 * \warning This is an internal function, use e_po3030k_ConfigCam
 * \param start The start row
 * \param stop The stop row
 * \param col The start/stop column
 * \return Zero if OK, non-zero if an error occur
```

```
 * \sa Datasheet p.42-43, e_po3030k_write_cam_registers and e_po3030k_config_cam
 */
int e_po3030k_set_vsync(unsigned int start,unsigned int stop,unsigned int col) {
unsigned char start_h,start_l,stop_h,stop_l,col_l,col_h;
if(start >= stop)
return -1;
if(stop > 499)
return -1;
if(col > 899)
return -1;
start_l = (unsigned char) start;
start_h = (unsigned char) (start >> 8);
stop_l = (unsigned char) stop;
stop_h = (unsigned char) (stop >> 8);
col_l = (unsigned char) col;
col_h = (unsigned char) (col >> 8);

cam_reg[VSYNCSTART_BASE] = start_h;
cam_reg[VSYNCSTART_BASE + 2] = start_l;

cam_reg[VSYNCSTOP_BASE] = stop_h;
cam_reg[VSYNCSTOP_BASE + 2] = stop_l;

cam_reg[VSYNCCOL_BASE] = col_h;
cam_reg[VSYNCCOL_BASE + 2] = col_l;

return 0;
}

#if PO3030K_FULL

/*! Set the register \a adr to value \a value
 * \param adr The address
 * \param value The value
 * \return 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
 * \sa e_po3030k_get_register
 */
int e_po3030k_set_register(unsigned char adr,unsigned char value) {
int i;
for(i = 0; i < 2*NB_REGISTERS; i+=2) {
if(cam_reg[i] == adr) {
cam_reg[i+1] = value;
return 0;
}
}
return -1;
}

/*! Get the register \a adr value
 * \param adr The address
 * \param value The pointer to the value to write to
 * \return 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
 * \sa e_po3030k_set_register
 */
int e_po3030k_get_register(unsigned char adr,unsigned char * value) {
int i;
for(i = 0; i < 2*NB_REGISTERS; i+=2) {
if(cam_reg[i] == adr) {
*value = cam_reg[i+1];
return 0;
}
}
return -1;
}

/*! Set the Pixel and amplificator bias
 * Increasing the bias produce better image quality, but increase camera's power consumption
 * \param pixbias The pixel bias
 * \param opbias The Amplificator bias
 * \sa Datasheet p.22
 */
void e_po3030k_set_bias(unsigned char pixbias, unsigned char opbias) {
cam_reg[BIAS_BASE] = opbias;
cam_reg[BIAS_BASE + 2] = pixbias;
}

/*! Set the gains of the camera
 * \param global The global gain \f$\in\lbrace 0,79 \rbrace\f$
 * \param red The red pixel's gain (fixed point [2:6] format)
 * \param green1 The green pixel near read one gain ([2:6] format)
 * \param green2 The green pixel near blue one gain ([2:6] format)
 * \param blue The blue pixel's gain ([2:6] format)
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.23-24
 */
int  e_po3030k_set_color_gain(unsigned char global, unsigned char red,
unsigned char green1, unsigned char green2,
unsigned char blue) {
if(global > 79)
return -1;

cam_reg[COLGAIN_BASE] = global;
cam_reg[COLGAIN_BASE + 2] = red;
cam_reg[COLGAIN_BASE + 4] = green1;
cam_reg[COLGAIN_BASE + 6] = blue;
cam_reg[COLGAIN_BASE + 8] = green2;
return 0;
}

/*! Set the pixel intergration time
 * This is counted in line-time interval. See dataseet p.25 for more information
 * \param time The integration time ( fixed point [14:6] format )
 * \sa Datasheet p.25
 */
void e_po3030k_set_integr_time(unsigned long time) {
```

```
cam_reg[INTEGR_BASE + 2] = (unsigned char) (time >> 6);
cam_reg[INTEGR_BASE] = (unsigned char) (time >> 14);
cam_reg[INTEGR_BASE + 4] = (unsigned char) (time << 8);
}

/*! Enable/Disable horizontal or vertical mirror
 * \param vertical Set to 1 when vertical mirror is enabled, 0 if disabled
 * \param horizontal Set to 1 when horizontal mirror is enabled, 0 if disabled
 * \sa Datasheet p.27
 */

void e_po3030k_set_mirror(int vertical, int horizontal) {
char val;
if(vertical && horizontal) {
val = 0b11001010;
} else {
if (vertical) {
val = 0b01001010;
} else {
if(horizontal) {
val = 0b10001010;
} else {
val = 0b00001010;
}
}
}
cam_reg[MIRROR_BASE] = val;
}

/*! Set the Analog to Digital Converter offset
 * \param offset The offset
 * \sa Datasheet p.28
 */
void e_po3030k_set_adc_offset(unsigned char offset) {
cam_reg[ADCOFF_BASE] = offset;
}

/*! Enable/Disable Sepia color
 * \param status Set \a status to 1 to enable, 0 to disable
 * \sa Datasheet p.34 and 74
 */
void e_po3030k_set_sepia(int status) {
if(status)
cam_reg[SEPIA_BASE] |= 0b10000000;
else
cam_reg[SEPIA_BASE] &= 0b01111111;
}

/*! Set lens shading gain
 * \param red Lens gain for red pixel \f$\in\lbrace 0,15 \rbrace\f$
 * \param green Lens gain for green pixel \f$\in\lbrace 0,15 \rbrace\f$
 * \param blue Lens gain for blue pixel \f$\in\lbrace 0,15 \rbrace\f$
 * \sa Datasheet p.36
 */
void e_po3030k_set_lens_gain(unsigned char red, unsigned char green, unsigned char blue) {
cam_reg[LENSG_BASE] = red;
cam_reg[LENSG_BASE + 2] = green;
cam_reg[LENSG_BASE + 4] = blue;
}

/*! Set Edge properties
 * \param gain Edge gain & moire factor (fixed point [2:3] format)
 * \param tresh Edge Enhancement threshold
 * \sa Datasheet p.36
 */

void e_po3030k_set_edge_prop(unsigned char gain, unsigned char tresh) {
cam_reg[EDGE_BASE] = 0b1010000 + (gain & 0b00011111);
cam_reg[EDGE_BASE + 2] = tresh;
}

/*! Set gamma coefficient
 * \warning This feature need extra care from the user
 * \param array Gamma coefficient array
 * \param color First two bytes : - 0b01 => Green
 * - 0b00 => Red
 * - else => Blue
 * \sa Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef
 */

void e_po3030k_set_gamma_coef(unsigned char array[12], char color) {
int i;
cam_reg[GAMMASELCOL_BASE] = 0b10000000 + (( color & 0x3 ) << 5);
for(i = 0; i < sizeof(array); i++)
cam_reg[GAMMA_BASE + i*2] = array[i];
}

/*! This special function write directly the Gamma coefficient and Gamma color select
 * into camera register.
 * \warning This function need extra care from the user
 * \sa e_po3030k_set_gamma_coef
 */

void e_po3030k_write_gamma_coef(void) {
e_i2cp_enable();
e_i2cp_write(DEVICE_ID,cam_reg[GAMMASELCOL_BASE - 1],cam_reg[GAMMASELCOL_BASE]);
e_i2cp_disable();
e_po3030k_sync_register_array(cam_reg[GAMMA_BASE - 1], cam_reg[GAMMA_BASE + 11 * 2 - 1]);
}

/*! 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
 * \param start The beginning address of the write
 * \param stop The last write address
 * \return The number of register written
 * \sa e_po3030k_write_cam_registers
```

```
 */

int  e_po3030k_sync_register_array(unsigned char start, unsigned char stop) {
int i;
int ret = 0;
e_i2cp_enable();
for(i=0;i < 2*NB_REGISTERS; i+=2 ) {
if(cam_reg[i] >= start && cam_reg[i] <= stop) {
e_i2cp_write(DEVICE_ID,cam_reg[i],cam_reg[i+1]);
ret++;
}
}

e_i2cp_disable();

return ret;
}

/*! Set color correction coefficient
 * \param array Color coefficient matrix (3x3), sign[7] | integer [6:5] | fractional [4:0]
 * \sa Datasheet p. 39
 */

void e_po3030k_SetColorMatrix(unsigned char array[3*3]) {
int i;
for(i = 0; i< sizeof(array); i++)
cam_reg[COLOR_COEF_BASE + i*2] = array[i];
}

/*! Set The color gain ( Cb/Cr )
 * \param cg11c Cb gain ( Sign[7] | Integer[6:5] | fractional[4:0] )
 * \param cg22c Cr gain ( Sign[7] | Integer[6:5] | fractional[4:0] )
 * \sa Datasheet p. 40
 */

void e_po3030k_set_cb_cr_gain(unsigned char cg11c, unsigned char cg22c) {
cam_reg[CBCRGAIN_BASE] = cg11c;
cam_reg[CBCRGAIN_BASE + 2] = cg22c;
}

/*! Set the Brightness & Contrast
 * \param bright The Brightness ( signed 1+7bits fixed point format )
 * \param contrast The Contrast
 * \sa Datasheet p. 40
 */

void e_po3030k_set_brigh_contr(unsigned char bright, unsigned char contrast) {
cam_reg[BRICTR_BASE] = bright;
cam_reg[BRICTR_BASE + 2] = contrast;
}

/*! Set The color tone at sepia color condition
 * \param cb Cb tone
 * \param cr Cr tone
 * \sa e_po3030k_set_sepia and Datasheet p. 41 and 74
 */

void e_po3030k_set_sepia_tone(unsigned char cb, unsigned char cr) {
cam_reg[SEPIATONE_BASE] = cb;
cam_reg[SEPIATONE_BASE + 2] = cr;
}

/*! Set the Center weight (Back Light compensation) Control parameter
 * \param ww Center weight ( 4 lower bits only )
 * \sa Datasheet p.44
 */

void e_po3030k_set_ww(unsigned char ww) {
cam_reg[WW_BASE] = (ww << 4) + 0b1111;
}

/*! Set AWB/AE tolerence margin
 * \param awbm AWV Margin ( 4 lower bits only )
 * \param aem AW Margin ( 4 lower bits only )
 * \sa Datasheet p.44
 */

void e_po3030k_set_awb_ae_tol(unsigned char awbm, unsigned char aem) {
cam_reg[AWVAETOL_BASE] = (awbm << 4) + (aem & 0b1111);
}

/*! Set AE speed
 * \param b AE speed factor when exposure time is decreasing ( 4 lower bits only )
 * \param d AE speed factor when exposure time is increasing ( 4 lower bits only )
 * \sa Datasheet p.44
 */

void e_po3030k_set_ae_speed(unsigned char b, unsigned char d) {
cam_reg[AESPEED_BASE] = (d << 4) + (b & 0b1111);
}

/*! Set exposure time
 * \param t Exposure time, LSB is in 1/64 line time
 * \warning Only writable if AE is disabled
 * \sa Datasheet p.45
 */

void e_po3030k_set_exposure(long t) {
cam_reg[EXPOSURE_BASE] = (unsigned char) (t >> 16);
cam_reg[EXPOSURE_BASE + 2] = (unsigned char) (t >> 8);
cam_reg[EXPOSURE_BASE + 4] = (unsigned char ) t;
}

/*! Set the reference exposure. The average brightness which the AE should have
 * \param exp The target exposure level
 * \sa  Datasheet p.45
 */
```

```
void e_po3030k_set_ref_exposure(unsigned char exp) {
cam_reg[REFREPO_BASE] = exp;
}

/*! Set the minimum and maximum exposure time in AE mode
 * \param min The minimum exposure time
 * \param max The maximum exposure time
 * \sa Datasheet p.46-47
 */

void e_po3030k_set_max_min_exp(unsigned int max, unsigned int min) {
cam_reg[MINMAXEXP_BASE] = (unsigned char) (max >> 8);
cam_reg[MINMAXEXP_BASE + 2] = (unsigned char) max;
cam_reg[MINMAXEXP_BASE + 4] = (unsigned char) (min >> 8);
cam_reg[MINMAXEXP_BASE + 6] = (unsigned char) min;
}

/*! Set the minimum and maximum red and blue gain in AWB mode
 * \param minb The minimum blue gain
 * \param maxb The maximum blue gain
 * \param minr The minimum red gain
 * \param maxr The maximum red gain
 * \param ratior The red gain ratio
 * \param ratiob The blue gain ratio
 * \sa Datasheet p. 47-48
 */
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) {
cam_reg[MINMAXAWB_BASE] = minr;
cam_reg[MINMAXAWB_BASE + 2] = maxr;
cam_reg[MINMAXAWB_BASE + 4] = minb;
cam_reg[MINMAXAWB_BASE + 6] = maxb;

cam_reg[MINMAXAWB_BASE + 8] = ratior;
cam_reg[MINMAXAWB_BASE + 10] = ratiob;
}

/*! Set the Weighting Window coordinate
 * \param x1 The X1 coordinate \f$\in\lbrace 211,x2 \rbrace\f$
 * \param x2 The X2 coordinate \f$\in\lbrace x1+1,423 \rbrace\f$
 * \param y1 The Y1 coordinate \f$\in\lbrace 160,y2 \rbrace\f$
 * \param y2 The Y2 coordinate \f$\in\lbrace y1+1,319 \rbrace\f$
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p. 49
 */
int  e_po3030k_set_weight_win(unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2) {
x1 += ARRAY_ORIGINE_X;
x2 += ARRAY_ORIGINE_X;
y1 += ARRAY_ORIGINE_Y;
y2 += ARRAY_ORIGINE_Y;

if(x2 > ARRAY_ORIGINE_X + ARRAY_WIDTH)
return -1;
if(y2 > ARRAY_ORIGINE_Y + ARRAY_HEIGHT)
return -1;

if(x1 >= x2 || y1 >= y2)
return -1;

if(x1 <= 421 || x2 >= 634 || y1 <= 167 || y2 >= 327)
return -1;

cam_reg[WEIGHWIN_BASE] = (unsigned char) (x1 >> 8);
cam_reg[WEIGHWIN_BASE + 2] = (unsigned char) x1;
cam_reg[WEIGHWIN_BASE + 4] = (unsigned char) (x2 >> 8);
cam_reg[WEIGHWIN_BASE + 6] = (unsigned char) x2;

cam_reg[WEIGHWIN_BASE + 8] = (unsigned char) (y1 >> 8);
cam_reg[WEIGHWIN_BASE + 10] = (unsigned char) y1;
cam_reg[WEIGHWIN_BASE + 12] = (unsigned char) (y2 >> 8);
cam_reg[WEIGHWIN_BASE + 14] = (unsigned char) y2;
return 0;
}

/*! Enable/Disable AWB and AE
 * \param awb 1 mean AWB enabled, 0 mean disabled
 * \param ae 1 mean AE enabled, 0 mean disabled
 * \sa Datasheet p. 50
 */

void e_po3030k_set_awb_ae(int awb, int ae) {
char val;
if(awb && ae) {
val = 0x3c;
} else {
if(awb) {
val = 0x2c;
} else {
if(ae) {
val = 0x1c;
} else {
val = 0x0c;
}
}
}
cam_reg[AWBAEENABLE_BASE] = val;
}

/*! Set flicker detection mode
 * \param manual Non-zero mean manual mode is enabled ( default automatic mode enabled )
 * \sa Datasheet p.29 e_po3030k_set_flicker_detection() e_po3030k_set_flicker_man_set()
 */

void e_po3030k_set_flicker_mode(int manual) {
if(manual) {
cam_reg[FLICKM_BASE] &= ~0x80;
```

```
} else {
cam_reg[FLICKM_BASE] |= 0x80;
}
}

/*! Set the 50/60Hz flicker detection
 * \param hz50 Non-zero mean 50Hz flicker detection enabled (default disabled)
 * \param 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.
 * \sa Datasheet p.29 e_po3030k_set_flicker_mode() e_po3030k_set_flicker_man_set()
 */

void e_po3030k_set_flicker_detection(int hz50, int hz60) {

if(hz50) {
cam_reg[FLICKM_BASE] &= ~0x40;
} else {
cam_reg[FLICKM_BASE] |= 0x40;
}

if(hz60) {
cam_reg[FLICKM_BASE] &= ~0x20;
} else {
cam_reg[FLICKM_BASE] |= 0x20;
}

}

/* Private internal function to get the real pixel clock */
static long po3030k_get_pixelclock(void) {
long clk;
switch (cam_reg[SPEED_ADDR]) {
case SPEED_2:
clk = MCLK/2;
break;
case SPEED_2_3:
clk = (MCLK*2)/3;
break;
case SPEED_4:
clk = MCLK/4;
break;
case SPEED_8:
clk = MCLK/8;
break;
case SPEED_16:
clk = MCLK/16;
break;
case SPEED_32:
clk = MCLK/32;
break;
case SPEED_64:
clk = MCLK/64;
break;
case SPEED_128:
clk = MCLK/128;
break;
default:
return 0;
}
if(cam_reg[COLOR_M_ADDR] == MODE_GRAYSCALE)
clk *= 2;
return clk;
}

/*! Set the camera's manual flicker's detection setting
 * \param hz50 The Hz for the 50Hz detection
 * \param hz60 The Hz for the 60Hz detection
 * \param fdm Flicker duration mode
 * \param fk Flicker count step
 * \param tol Flicker tolerance
 * \warning You must have set the mode ( image size, color ) before calling this function
 * \return Non-zero if an error occur, 0 if OK
 * \sa Datasheet p.29-30 e_po3030k_set_flicker_detection() e_po3030k_set_flicker_mode()
 */

int  e_po3030k_set_flicker_man_set(int hz50, int hz60, int fdm, int fk, int tol) {
unsigned int p50, p60;
/* checking if parameter are correct */
if(fk < 0 || fk > 3)
return -1;

if(tol < 0 || tol > 3)
return -1;

/* The datasheet isn't clear about this formula
 * At p.30 it's written:
 * Period = PixelClock / (HZ * 2 * FrameWidth)
 * But at p.60 it's written:
 * Period = 256*(MasterClock /(FrameWidth * 2))/(HZ*2)
   *        = 64*MasterClock / (FrameWidth * HZ)
 */

p50 = ( po3030k_get_pixelclock() * 2 * FRAME_WIDTH) / (long) hz50;
p60 = ( po3030k_get_pixelclock() * 2 * FRAME_WIDTH) / (long) hz60;

cam_reg[FLICKP_BASE] = (unsigned char) (p50 >> 8);
cam_reg[FLICKP_BASE + 2] = (unsigned char) p50;
cam_reg[FLICKP_BASE + 4] = (unsigned char) (p60 >> 8);
cam_reg[FLICKP_BASE + 6] = (unsigned char) p60;

if(fdm) {
cam_reg[FLICKM_BASE] |= 0x10;
} else {
cam_reg[FLICKM_BASE] &= ~0x10;
}
```

```
cam_reg[FLICKM_BASE] = (tol & 0x3) + ((fk & 0x3) << 2) + (cam_reg[FLICKM_BASE] & 0xF0);

return 0;
}
#endif
```

2.4.5 e_timers.c

```
/*! \file
 * \ingroup cameral
 * \brief Manage camera's interrupts (two timers)
 * \author Philippe Rtornaz
 * \verbinclude interrupt.s
 */


#include <p30f6014A.h>
#include "../../motor_led/e_epuck_ports.h"
#include "e_po3030k.h"



/*! The buffer to write to */
char * _po3030k_buffer;

/*! The flag to tell, the image is ready or not
 * Zero mean capture is in progress, non-zero mean capture done.
 * \sa e_po3030k_is_img_ready
 */
int _po3030k_img_ready;

static int blank_row_betw;

int _po3030k_current_row;
int _po3030k_row;

char _po3030k_line_conf[330];

/*! \brief 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 )
 */
void __attribute__((interrupt, auto_psv))
_T5Interrupt(void) {
IFS1bits.T5IF = 0;
/* let's enable Hsync */
T4CONbits.TON = 1;
/* single shot */
T5CONbits.TON = 0;
}


static void init_timer5(void) {
/* external pin, 1:1 prescaler */
T5CON = 0x2;
TMR5 = 0;
PR5 = 1;
IFS1bits.T5IF = 0;
IEC1bits.T5IE = 1;
T5CONbits.TON = 1;
}

static void init_timer4(void) {
T4CON = 0x2;
TMR4 = blank_row_betw;
PR4 = blank_row_betw + 1;
IFS1bits.T4IF = 0;
T4CONbits.TON = 0;
IEC1bits.T4IE = 1;
}



/*! Launch a capture in the \a buf buffer
 * \param buf The buffer to write to
 * \sa e_po3030k_config_cam and e_po3030k_is_img_ready
 */
void e_po3030k_launch_capture(char * buf) {
_po3030k_current_row = 0;
_po3030k_buffer = buf;
_po3030k_img_ready = 0;
init_timer4();
/* Timer5 must ALWAY be initialized as the last one */
init_timer5();

}

/*! Modify the interrupt configuration
 * \warning This is an internal function, use \a e_po3030k_config_cam
 * \param pixel_row The number of row to take
 * \param pixel_col The number of pixel to take each \a pixel_row
 * \param bpp The number of byte per pixel
 * \param pbp The number of pixel to ignore between each pixel
 * \param bbl The number of row to ignore between each line
 * \return Zero if OK, non-zero if the mode exceed internal data representation
 * \sa e_po3030k_get_bytes_per_pixel and e_po3030k_config_cam
 */
int e_po3030k_apply_timer_config(int pixel_row, int pixel_col, int bpp, int pbp, int bbl) {
int i;
int pos = 0;
if(pixel_col * bpp * (1+pbp) + 1 > sizeof(_po3030k_line_conf))
return -1;

for(i = 0; i < pixel_col; i++) {
```

```
int j;
for(j = 0; j < bpp; j++) {
_po3030k_line_conf[pos++] = 1;
}
for(j = 0; j < pbp*bpp; j++)  {
_po3030k_line_conf[pos++] = 0;
}
}
_po3030k_line_conf[pos] = 2;  /* flag to tell "line end here" */
blank_row_betw = bbl;
_po3030k_row = pixel_row;

return 0;
}

/*! Check if the current capture is finished
 * \return Zero if the current capture is in progress, non-zero if the capture is done.
 * \sa e_po3030k_launch_capture
 */
int e_po3030k_is_img_ready(void) {
return _po3030k_img_ready;
}
```

2.5 camera/slow_3_timer

2.5.1 e_calc.c

```
/*! \file
 * \ingroup camera2
 * \brief 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"

#define  ARRAY_ORIGINE_X 210
#define ARRAY_ORIGINE_Y 7


/*! 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.
 * \param sensor_x1 The X coordinate of the window's corner
 * \param sensor_y1 The Y coordinate of the window's corner
 * \param sensor_width the Width of the interest area, in FULL sampling scale
 * \param sensor_height The Height of the insterest area, in FULL sampling scale
 * \param zoom_fact_width The subsampling to apply for the window's Width
 * \param zoom_fact_height The subsampling to apply for the window's Height
 * \param color_mode The color mode in which the camera should be configured
 * \return Zero if the settings are correct, non-zero if an error occur
 * \sa e_po3030k_write_cam_registers
 */

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 pbp_h, pbp_w;
int nb_pixels, nb_lines;
int real_zoom_h, real_zoom_w;
int sampl_mode;
sensor_x1 += ARRAY_ORIGINE_X;
sensor_y1 += ARRAY_ORIGINE_Y;
/* testing if the mode is acceptable */
if(zoom_fact_height < 1 || zoom_fact_width < 1)
return -1;

/* Check if the area is out of bound */
if((sensor_x1 + sensor_width) > (ARRAY_ORIGINE_X + ARRAY_WIDTH))
return -1;
if((sensor_y1 + sensor_height) > (ARRAY_ORIGINE_Y + ARRAY_HEIGHT))
return -1;

/* Check if Sensor[Width|Height] is a multiple of Zoom */
if(sensor_width % zoom_fact_width)
return -1;
if(sensor_height % zoom_fact_height)
return -1;

/* Search the best subsampling aviable */
if(!(zoom_fact_height%4)) {
if(!(zoom_fact_width%4)) {
sampl_mode = MODE_QQVGA;
real_zoom_h = zoom_fact_height >> 2;
real_zoom_w = zoom_fact_width >> 2;
/* this is done because we must have Vsync an effective
 * row before the real picture */
sensor_y1 -= 4;
} else {
if(!(zoom_fact_width%2)) {
sampl_mode = MODE_QVGA;
real_zoom_h = zoom_fact_height >> 1;
real_zoom_w = zoom_fact_width >> 1;
sensor_y1 -= 2;
} else {
sampl_mode = MODE_VGA;
real_zoom_h = zoom_fact_height;
real_zoom_w = zoom_fact_width;
sensor_y1--;
```

```
}
}
} else if(!(zoom_fact_height%2)) {
if(!(zoom_fact_width%2)) {
sampl_mode = MODE_QVGA;
real_zoom_h = zoom_fact_height >> 1;
real_zoom_w = zoom_fact_width >> 1;
sensor_y1 -= 2;
} else {
sampl_mode = MODE_VGA;
real_zoom_h = zoom_fact_height;
real_zoom_w = zoom_fact_width;
sensor_y1--;
}
} else {
sampl_mode = MODE_VGA;
real_zoom_w = zoom_fact_width;
real_zoom_h = zoom_fact_height;
sensor_y1--;
}
pbp_w = real_zoom_w - 1;
pbp_h = real_zoom_h - 1;

nb_pixels = sensor_width / zoom_fact_width;
nb_lines = sensor_height/ zoom_fact_height;

/* set camera configuration */
if(e_po3030k_set_color_mode(color_mode))
return -1;

if(color_mode == GREY_SCALE_MODE) {
switch (sampl_mode) {
case MODE_VGA:
e_po3030k_set_speed(SPEED_64);
break;
case MODE_QVGA:
e_po3030k_set_speed(SPEED_32);
break;
case MODE_QQVGA:
e_po3030k_set_speed(SPEED_16);
break;
}
} else {
switch (sampl_mode) {
case MODE_VGA:
e_po3030k_set_speed(SPEED_128);
break;
case MODE_QVGA:
e_po3030k_set_speed(SPEED_64);
break;
case MODE_QQVGA:
e_po3030k_set_speed(SPEED_32);
break;
}
}

if(e_po3030k_set_sampling_mode(sampl_mode))
return -1;

if(e_po3030k_set_wx(sensor_x1,ARRAY_ORIGINE_X + ARRAY_WIDTH + 1))
return -1;

if(e_po3030k_set_wy(sensor_y1,ARRAY_ORIGINE_Y + ARRAY_HEIGHT + 1))
return -1;

if(e_po3030k_set_vsync(sensor_y1 - 1,ARRAY_ORIGINE_Y + ARRAY_HEIGHT, sensor_x1 + 1))
return -1;

/* set timer configuration */
e_po3030k_apply_timer_config(nb_lines,nb_pixels,e_po3030k_get_bytes_per_pixel(color_mode),pbp_w,pbp_h);
return 0;

}

/*! Return the number of bytes per pixel in the given color mode
 * \param color_mode The given color mode
 * \return The number of bytes per pixel in the given color mode
 */
int e_po3030k_get_bytes_per_pixel(int color_mode) {
switch (color_mode) {
case GREY_SCALE_MODE:
return 1;
case RGB_565_MODE:
return 2;
case YUV_MODE:
return 2; /* YUV mode is not the best mode for subsampling */
}
/* UNKNOW ! */
return 1;
}

/*! Initialize the camera, must be called before any other function
 */
void e_po3030k_init_cam(void) {
int i;
e_init_port();
e_i2cp_init();
CAM_RESET=0;
for(i=100;i;i--) __asm__ volatile ("nop");
CAM_RESET=1;
for(i=100;i;i--) __asm__ volatile ("nop");
}
```

2.5.2 e_po3030k.h

```
/*! \file
 * \ingroup camera2
 * \brief PO3030k library header (three timers)
 * \author Philippe Rtornaz
 */

/*! \defgroup camera2 Camera slow three timers
 *
 * \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.
 *
 * \section intro_sec 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(), 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().
 * This is typically done after every configuration call and before acquire
 * the first picture.
 *
 * \section defset 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.
 *
 * \section perfsec 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
 *
 * \section exemple_sect Exemples
 *
 * \subsection ex1_sect Basic exemple
 *
 * \code
#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;
}
\endcode

 * 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.
 *
 * \subsection ex2_sect More advanced exemple
\code
#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;
}
\endcode
 * 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.
 */
```

```
#ifndef __PO3030K_H__
#define __PO3030K_H__

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



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);

#if PO3030K_FULL

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);

#endif

#endif
```

2.5.3 e_registers.c

```
/*! \file
 * \ingroup camera2
 * \brief Manage po3030k registers (three timers)
 * \author Philippe Rtornaz
 */



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

#define MCLK ((long) 14745600)  /* 14.7456Mhz */
#define MCLK_P_NS 0.067816840278 /* Master clock period in ns */


/* these two define should go into a private header but, well,
 * I won't do it one for only two define ... */
#define  ARRAY_ORIGINE_X 210
#define ARRAY_ORIGINE_Y 7

/* Some preprocessor ugliness */
#if PO3030K_FULL
#define BASE_D1 0
#define BASE_D2 0
#define BASE_D3 0
#define BASE_D4 0
#else
#define BASE_D1 20
#define BASE_D2 (BASE_D1 + 4)
#define BASE_D3 (BASE_D2 + 8)
#define BASE_D4 (BASE_D3 + 76)
#endif

#define DEVICE_ID   0xDC

static unsigned char cam_reg[] = {
/* format:
   address,   value */
#define FRAME_WIDTH 0x0353
0x04,(unsigned char) (FRAME_WIDTH >> 8),
0x05,(unsigned char) FRAME_WIDTH,
#define FRAME_HEIGTH 0x01e8
0x06,(unsigned char) (FRAME_WIDTH >> 8),
0x07,(unsigned char) FRAME_WIDTH,
#define WINDOW_X1_BASE 9
0x08,0x00,
0x09,0xd1,
#define WINDOW_Y1_BASE 13
0x0a,0x00,
0x0b,0x07,
#define WINDOW_X2_BASE 17
0x0c,0x03,
0x0d,0x51,
#define WINDOW_Y2_BASE 21
0x0e,0x01,
0x0f,0xe7,

#if PO3030K_FULL
#define BIAS_BASE 25
0x12,0x02,
0x13,0x02,
#define COLGAIN_BASE 29
0x15,0x10,
0x16,0x40,
0x17,0x40,
0x18,0x40,
0x19,0x40,
#define INTEGR_BASE 39
0x1a,0x03,
0x1b,0xe6,
0x1c,0x00,
#endif

#define SPEED_ADDR (45 - BASE_D1)
0x1d,/*0x00*/ 0b01110000,

#if PO3030K_FULL
#define MIRROR_BASE 47
0x1e,0x0a,
0x1f,0x1b,
#endif
```

```
#define SAMPLING_ADDR ( 51 - BASE_D2 )
0x20,MODE_VGA,
#define ADCOFF_BASE ( 53 - BASE_D2 )
0x3b,0x00,
#define FLICKM_BASE ( 55 - BASE_D2 )
0x46,0x80,
/* see datasheet p. 30 */
0x48,(unsigned char) (1667./(MCLK_P_NS*256)),

#if PO3030K_FULL
#define FLICKP_BASE 59
0x49,0x96,
0x4a,0x00,
0x4b,0x7d,
0x4c,0x00,
#endif
#define COLOR_M_ADDR (67 - BASE_D3)
#define MODE_R5G6B5 0x08
#define MODE_YUV 0x02
#define MODE_GRAYSCALE 0x0c
0x4e,MODE_R5G6B5,

#if PO3030K_FULL
#define SEPIA_BASE 69
0x4f,0x0a,
0x50,0x70,
#define LENSG_BASE 73
0x59,0x00,
0x5a,0x00,
0x5b,0x00,
#define EDGE_BASE 79
0x5c,0xac,
0x5d,0x02,
#define GAMMA_BASE 83
0x76,0x00,
0x77,0x1a,
0x78,0x2a,
0x79,0x37,
0x7a,0x42,
0x7b,0x56,
0x7c,0x68,
0x7d,0x87,
0x7e,0xa3,
0x7f,0xbc,
0x80,0xd4,
0x81,0xea,
#define COLOR_COEF_BASE 107
0x8e,0x38,
0x8f,0xa5,
0x90,0x0d,
0x91,0x93,
0x92,0x2d,
0x93,0x06,
0x94,0x83,
0x95,0xaa,
0x96,0x4d,
#define CBCRGAIN_BASE 125
0x97,0x25,
0x98,0x25,
#define BRICTR_BASE 129
0x9b,0x00,
0x9c,0x96,
#define SEPIATONE_BASE 133
0x9f,0x80,
0xa0,0x80,
0xa1,0xb6,
0xa2,0x9d,
0xa3,0xab,
0xa4,0x80,
#endif

#define VSYNCSTART_BASE (145 - BASE_D4)
0xa5,0x00,
0xa6,0x07,
#define VSYNCSTOP_BASE (149 - BASE_D4)
0xa7,0x01,
0xa8,0xe7,
#define VSYNCCOL_BASE (153 - BASE_D4)
0xa9,0x00,
0xaa,0x01,

#if PO3030K_FULL
#define WW_BASE 157
0xad,0x9f,
#define AWVAETOL_BASE 159
0xae,0x22,
#define AESPEED_BASE 161
0xaf,0x8c,
#define EXPOSURE_BASE 163
0xb0,0x07,
0xb1,0xcc,
0xb2,0x00,
#define REFREPO_BASE 169
0xb3,0x70,
0xb9,0x03,
0xba,0xe6,
#define MINMAXEXP_BASE 175
0xbb,0x07,
0xbc,0xcc,
0xbd,0x00,
0xbe,0x0c,
#define MINMAXAWB_BASE 183
0xc1,0x10,
0xc2,0xe0,
0xc3,0x10,
0xc4,0xe0,
0xc5,0x80,
```

```
0xc6,0x80,
#define WEIGHWIN_BASE 195
0xc7,0x01,
0xc8,0xa5,
0xc9,0x02,
0xca,0x7a,
0xcb,0x00,
0xcc,0xa7,
0xcd,0x01,
0xce,0x47,
#define AWBAEENABLE_BASE 211
0xd4,0x3c,
0xd5,0x01,
#define GAMMASELCOL_BASE 215
0xd8,0x80
#endif
};
#define NB_REGISTERS (sizeof(cam_reg)/(2*sizeof(cam_reg[0])))


/*! 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.
 * \sa e_po3030k_read_cam_registers
 */
void e_po3030k_write_cam_registers(void) {
int i;
e_i2cp_enable();
for(i=0;i < 2*NB_REGISTERS; i+=2 )
e_i2cp_write(DEVICE_ID,cam_reg[i],cam_reg[i+1]);

e_i2cp_disable();
}

/*! Read the camera register
 * \sa e_po3030k_write_cam_registers
 */
void e_po3030k_read_cam_registers(void) {
int i;
e_i2cp_enable();
for(i=0;i < 2*NB_REGISTERS; i+=2 )
cam_reg[i+1] = e_i2cp_read(DEVICE_ID,cam_reg[i]);
e_i2cp_disable();
}

/*! Set the camera color mode
 * \warning This is an internal function, use e_po3030k_config_cam
 * \param mode The color mode
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p. 31, e_po3030k_write_cam_registers and e_po3030k_config_cam
 */
int e_po3030k_set_color_mode(int mode) {
switch (mode) {
case GREY_SCALE_MODE:
cam_reg[COLOR_M_ADDR] = MODE_GRAYSCALE;
break;
case RGB_565_MODE:
cam_reg[COLOR_M_ADDR] = MODE_R5G6B5;
break;
case YUV_MODE:
cam_reg[COLOR_M_ADDR] = MODE_YUV;
break;
default:
return -1;
}
return 0;
}

/*! Set the camera sampling mode
 * \warning This is an internal function, use e_po3030k_config_cam
 * \param mode The given sampling mode
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p. 28 and e_po3030k_config_cam
 */
int e_po3030k_set_sampling_mode(int mode) {
if(mode == MODE_VGA || mode == MODE_QVGA ||
mode == MODE_QQVGA )
{
cam_reg[SAMPLING_ADDR] = mode;
return 0;
}
return -1;
}

/*! Set the camera speed
 * \warning This is an internal function, use e_po3030k_config_cam
 * \param mode The given speed
 * \return Zero if OK, non-zero if unknow mode
 * \sa Datasheet p. 26 and e_po3030k_config_cam
 */

int e_po3030k_set_speed(int mode) {
if(mode == SPEED_2 || mode == SPEED_4 ||
mode == SPEED_8 || mode == SPEED_16 ||
mode == SPEED_32 || mode == SPEED_64 ||
mode == SPEED_64 || mode == SPEED_2_3)
{
cam_reg[SPEED_ADDR] = mode;
return 0;
}

return -1;
}
/*! Set the camera window X coordinate
 * \warning This is an internal function, use e_po3030k_ConfigCam
 * \param start The start column
```

```
 * \param stop The stop column
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.20-21, e_po3030k_write_cam_registers, e_po3030k_set_wy and e_po3030k_config_cam
 */
int e_po3030k_set_wx(unsigned int start, unsigned int stop) {
unsigned char start_h,start_l,stop_h,stop_l;
if(start >= stop)
return -1;
if(stop > 899)
return -1;
start--;
start_l = (unsigned char) start;
start_h = (unsigned char) (start >> 8);
stop_l = (unsigned char) stop;
stop_h = (unsigned char) (stop >> 8);

cam_reg[WINDOW_X1_BASE] = start_h;
cam_reg[WINDOW_X1_BASE + 2] = start_l;

cam_reg[WINDOW_X2_BASE] = stop_h;
cam_reg[WINDOW_X2_BASE + 2] = stop_l;

return 0;
}

/*! Set the camera window Y coordinate
 * \warning This is an internal function, use e_po3030k_ConfigCam
 * \param start The start row
 * \param stop The stop row
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam
 */
int e_po3030k_set_wy(unsigned int start, unsigned int stop) {
unsigned char start_h,start_l,stop_h,stop_l;
if(start >= stop)
return -1;
if(stop > 499)
return -1;
start_l = (unsigned char) start;
start_h = (unsigned char) (start >> 8);
stop_l = (unsigned char) stop;
stop_h = (unsigned char) (stop >> 8);

cam_reg[WINDOW_Y1_BASE] = start_h;
cam_reg[WINDOW_Y1_BASE + 2] = start_l;

cam_reg[WINDOW_Y2_BASE] = stop_h;
cam_reg[WINDOW_Y2_BASE + 2] = stop_l;

return 0;
}

/*! Set the camera window VSYNC coordinate
 * \warning This is an internal function, use e_po3030k_ConfigCam
 * \param start The start row
 * \param stop The stop row
 * \param col The start/stop column
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.42-43, e_po3030k_write_cam_registers and e_po3030k_config_cam
 */
int e_po3030k_set_vsync(unsigned int start,unsigned int stop,unsigned int col) {
unsigned char start_h,start_l,stop_h,stop_l,col_l,col_h;
if(start >= stop)
return -1;
if(stop > 499)
return -1;
if(col > 899)
return -1;
start_l = (unsigned char) start;
start_h = (unsigned char) (start >> 8);
stop_l = (unsigned char) stop;
stop_h = (unsigned char) (stop >> 8);
col_l = (unsigned char) col;
col_h = (unsigned char) (col >> 8);

cam_reg[VSYNCSTART_BASE] = start_h;
cam_reg[VSYNCSTART_BASE + 2] = start_l;

cam_reg[VSYNCSTOP_BASE] = stop_h;
cam_reg[VSYNCSTOP_BASE + 2] = stop_l;

cam_reg[VSYNCCOL_BASE] = col_h;
cam_reg[VSYNCCOL_BASE + 2] = col_l;

return 0;
}

#if PO3030K_FULL

/*! Set the register \a adr to value \a value
 * \param adr The address
 * \param value The value
 * \return 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
 * \sa e_po3030k_get_register
 */
int e_po3030k_set_register(unsigned char adr,unsigned char value) {
int i;
for(i = 0; i < 2*NB_REGISTERS; i+=2) {
if(cam_reg[i] == adr) {
cam_reg[i+1] = value;
return 0;
}
}
return -1;
}
```

```
/*! Get the register \a adr value
 * \param adr The address
 * \param value The pointer to the value to write to
 * \return 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
 * \sa e_po3030k_set_register
 */
int e_po3030k_get_register(unsigned char adr,unsigned char * value) {
int i;
for(i = 0; i < 2*NB_REGISTERS; i+=2) {
if(cam_reg[i] == adr) {
*value = cam_reg[i+1];
return 0;
}
}
return -1;
}

/*! Set the Pixel and amplificator bias
 * Increasing the bias produce better image quality, but increase camera's power consumption
 * \param pixbias The pixel bias
 * \param opbias The Amplificator bias
 * \sa Datasheet p.22
 */
void e_po3030k_set_bias(unsigned char pixbias, unsigned char opbias) {
cam_reg[BIAS_BASE] = opbias;
cam_reg[BIAS_BASE + 2] = pixbias;
}

/*! Set the gains of the camera
 * \param global The global gain \f$\in\lbrace 0,79 \rbrace\f$
 * \param red The red pixel's gain (fixed point [2:6] format)
 * \param green1 The green pixel near read one gain ([2:6] format)
 * \param green2 The green pixel near blue one gain ([2:6] format)
 * \param blue The blue pixel's gain ([2:6] format)
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p.23-24
 */
int  e_po3030k_set_color_gain(unsigned char global, unsigned char red,
unsigned char green1, unsigned char green2,
unsigned char blue) {
if(global > 79)
return -1;

cam_reg[COLGAIN_BASE] = global;
cam_reg[COLGAIN_BASE + 2] = red;
cam_reg[COLGAIN_BASE + 4] = green1;
cam_reg[COLGAIN_BASE + 6] = blue;
cam_reg[COLGAIN_BASE + 8] = green2;
return 0;
}

/*! Set the pixel intergration time
 * This is counted in line-time interval. See dataseet p.25 for more information
 * \param time The integration time ( fixed point [14:6] format )
 * \sa Datasheet p.25
 */
void e_po3030k_set_integr_time(unsigned long time) {
cam_reg[INTEGR_BASE + 2] = (unsigned char) (time >> 6);
cam_reg[INTEGR_BASE] = (unsigned char) (time >> 14);
cam_reg[INTEGR_BASE + 4] = (unsigned char) (time << 8);
}

/*! Enable/Disable horizontal or vertical mirror
 * \param vertical Set to 1 when vertical mirror is enabled, 0 if disabled
 * \param horizontal Set to 1 when horizontal mirror is enabled, 0 if disabled
 * \sa Datasheet p.27
 */

void e_po3030k_set_mirror(int vertical, int horizontal) {
char val;
if(vertical && horizontal) {
val = 0b11001010;
} else {
if (vertical) {
val = 0b01001010;
} else {
if(horizontal) {
val = 0b10001010;
} else {
val = 0b00001010;
}
}
}
cam_reg[MIRROR_BASE] = val;
}

/*! Set the Analog to Digital Converter offset
 * \param offset The offset
 * \sa Datasheet p.28
 */
void e_po3030k_set_adc_offset(unsigned char offset) {
cam_reg[ADCOFF_BASE] = offset;
}

/*! Enable/Disable Sepia color
 * \param status Set \a status to 1 to enable, 0 to disable
 * \sa Datasheet p.34 and 74
 */
void e_po3030k_set_sepia(int status) {
if(status)
cam_reg[SEPIA_BASE] |= 0b10000000;
else
cam_reg[SEPIA_BASE] &= 0b01111111;
}

/*! Set lens shading gain
```

```
 * \param red Lens gain for red pixel \f$\in\lbrace 0,15 \rbrace\f$
 * \param green Lens gain for green pixel \f$\in\lbrace 0,15 \rbrace\f$
 * \param blue Lens gain for blue pixel \f$\in\lbrace 0,15 \rbrace\f$
 * \sa Datasheet p.36
 */
void e_po3030k_set_lens_gain(unsigned char red, unsigned char green, unsigned char blue) {
cam_reg[LENSG_BASE] = red;
cam_reg[LENSG_BASE + 2] = green;
cam_reg[LENSG_BASE + 4] = blue;
}

/*! Set Edge properties
 * \param gain Edge gain & moire factor (fixed point [2:3] format)
 * \param tresh Edge Enhancement threshold
 * \sa Datasheet p.36
 */

void e_po3030k_set_edge_prop(unsigned char gain, unsigned char tresh) {
cam_reg[EDGE_BASE] = 0b10100000 + (gain & 0b00011111);
cam_reg[EDGE_BASE + 2] = tresh;
}

/*! Set gamma coefficient
 * \warning This feature need extra care from the user
 * \param array Gamma coefficient array
 * \param color First two bytes : - 0b01 => Green
 * - 0b00 => Red
 * - else => Blue
 * \sa Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef
 */

void e_po3030k_set_gamma_coef(unsigned char array[12], char color) {
int i;
cam_reg[GAMMASELCOL_BASE] = 0b10000000 + (( color & 0x3 ) << 5);
for(i = 0; i < sizeof(array); i++)
cam_reg[GAMMA_BASE + i*2] = array[i];
}

/*! This special function write directly the Gamma coefficient and Gamma color select
 * into camera register.
 * \warning This function need extra care from the user
 * \sa e_po3030k_set_gamma_coef
 */

void e_po3030k_write_gamma_coef(void) {
e_i2cp_enable();
e_i2cp_write(DEVICE_ID,cam_reg[GAMMASELCOL_BASE - 1],cam_reg[GAMMASELCOL_BASE]);
e_i2cp_disable();
e_po3030k_sync_register_array(cam_reg[GAMMA_BASE - 1], cam_reg[GAMMA_BASE + 11 * 2 - 1]);
}

/*! 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
 * \param start The beginning address of the write
 * \param stop The last write address
 * \return The number of register written
 * \sa e_po3030k_write_cam_registers
 */

int  e_po3030k_sync_register_array(unsigned char start, unsigned char stop) {
int i;
int ret = 0;
e_i2cp_enable();
for(i=0;i < 2*NB_REGISTERS; i+=2 ) {
if(cam_reg[i] >= start && cam_reg[i] <= stop) {
e_i2cp_write(DEVICE_ID,cam_reg[i],cam_reg[i+1]);
ret++;
}
}

e_i2cp_disable();

return ret;
}

/*! Set color correction coefficient
 * \param array Color coefficient matrix (3x3), sign[7] | integer [6:5] | fractional [4:0]
 * \sa Datasheet p. 39
 */

void e_po3030k_SetColorMatrix(unsigned char array[3*3]) {
int i;
for(i = 0; i< sizeof(array); i++)
cam_reg[COLOR_COEF_BASE + i*2] = array[i];
}

/*! Set The color gain ( Cb/Cr )
 * \param cg11c Cb gain ( Sign[7] | Integer[6:5] | fractional[4:0] )
 * \param cg22c Cr gain ( Sign[7] | Integer[6:5] | fractional[4:0] )
 * \sa Datasheet p. 40
 */

void e_po3030k_set_cb_cr_gain(unsigned char cg11c, unsigned char cg22c) {
cam_reg[CBCRGAIN_BASE] = cg11c;
cam_reg[CBCRGAIN_BASE + 2] = cg22c;
}

/*! Set the Brightness & Contrast
 * \param bright The Brightness ( signed 1+7bits fixed point format )
 * \param contrast The Contrast
 * \sa Datasheet p. 40
 */

void e_po3030k_set_brigh_contr(unsigned char bright, unsigned char contrast) {
cam_reg[BRICTR_BASE] = bright;
cam_reg[BRICTR_BASE + 2] = contrast;
```

```
}

/*! Set The color tone at sepia color condition
 * \param cb Cb tone
 * \param cr Cr tone
 * \sa e_po3030k_set_sepia and Datasheet p. 41 and 74
 */

void e_po3030k_set_sepia_tone(unsigned char cb, unsigned char cr) {
cam_reg[SEPIATONE_BASE] = cb;
cam_reg[SEPIATONE_BASE + 2] = cr;
}

/*! Set the Center weight (Back Light compensation) Control parameter
 * \param ww Center weight ( 4 lower bits only )
 * \sa Datasheet p.44
 */

void e_po3030k_set_ww(unsigned char ww) {
cam_reg[WW_BASE] = (ww << 4) + 0b1111;
}

/*! Set AWB/AE tolerence margin
 * \param awbm AWV Margin ( 4 lower bits only )
 * \param aem AW Margin ( 4 lower bits only )
 * \sa Datasheet p.44
 */

void e_po3030k_set_awb_ae_tol(unsigned char awbm, unsigned char aem) {
cam_reg[AWVAETOL_BASE] = (awbm << 4) + (aem & 0b1111);
}

/*! Set AE speed
 * \param b AE speed factor when exposure time is decreasing ( 4 lower bits only )
 * \param d AE speed factor when exposure time is increasing ( 4 lower bits only )
 * \sa Datasheet p.44
 */

void e_po3030k_set_ae_speed(unsigned char b, unsigned char d) {
cam_reg[AESPEED_BASE] = (d << 4) + (b & 0b1111);
}

/*! Set exposure time
 * \param t Exposure time, LSB is in 1/64 line time
 * \warning Only writable if AE is disabled
 * \sa Datasheet p.45
 */

void e_po3030k_set_exposure(long t) {
cam_reg[EXPOSURE_BASE] = (unsigned char) (t >> 16);
cam_reg[EXPOSURE_BASE + 2] = (unsigned char) (t >> 8);
cam_reg[EXPOSURE_BASE + 4] = (unsigned char ) t;
}

/*! Set the reference exposure. The average brightness which the AE should have
 * \param exp The target exposure level
 * \sa  Datasheet p.45
 */

void e_po3030k_set_ref_exposure(unsigned char exp) {
cam_reg[REFREPO_BASE] = exp;
}

/*! Set the minimum and maximum exposure time in AE mode
 * \param min The minimum exposure time
 * \param max The maximum exposure time
 * \sa Datasheet p.46-47
 */

void e_po3030k_set_max_min_exp(unsigned int max, unsigned int min) {
cam_reg[MINMAXEXP_BASE] = (unsigned char) (max >> 8);
cam_reg[MINMAXEXP_BASE + 2] = (unsigned char) max;
cam_reg[MINMAXEXP_BASE + 4] = (unsigned char) (min >> 8);
cam_reg[MINMAXEXP_BASE + 6] = (unsigned char) min;
}

/*! Set the minimum and maximum red and blue gain in AWB mode
 * \param minb The minimum blue gain
 * \param maxb The maximum blue gain
 * \param minr The minimum red gain
 * \param maxr The maximum red gain
 * \param ratior The red gain ratio
 * \param ratiob The blue gain ratio
 * \sa Datasheet p. 47-48
 */
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) {
cam_reg[MINMAXAWB_BASE] = minr;
cam_reg[MINMAXAWB_BASE + 2] = maxr;
cam_reg[MINMAXAWB_BASE + 4] = minb;
cam_reg[MINMAXAWB_BASE + 6] = maxb;

cam_reg[MINMAXAWB_BASE + 8] = ratior;
cam_reg[MINMAXAWB_BASE + 10] = ratiob;
}

/*! Set the Weighting Window coordinate
 * \param x1 The X1 coordinate \f$\in\lbrace 211,x2 \rbrace\f$
 * \param x2 The X2 coordinate \f$\in\lbrace x1+1,423 \rbrace\f$
 * \param y1 The Y1 coordinate \f$\in\lbrace 160,y2 \rbrace\f$
 * \param y2 The Y2 coordinate \f$\in\lbrace y1+1,319 \rbrace\f$
 * \return Zero if OK, non-zero if an error occur
 * \sa Datasheet p. 49
 */
int  e_po3030k_set_weight_win(unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2) {
x1 += ARRAY_ORIGINE_X;
x2 += ARRAY_ORIGINE_X;
```

```
y1 += ARRAY_ORIGINE_Y;
y2 += ARRAY_ORIGINE_Y;

if(x2 > ARRAY_ORIGINE_X + ARRAY_WIDTH)
return -1;
if(y2 > ARRAY_ORIGINE_Y + ARRAY_HEIGHT)
return -1;

if(x1 >= x2 || y1 >= y2)
return -1;

if(x1 <= 421 || x2 >= 634 || y1 <= 167 || y2 >= 327)
return -1;

cam_reg[WEIGHWIN_BASE] = (unsigned char) (x1 >> 8);
cam_reg[WEIGHWIN_BASE + 2] = (unsigned char) x1;
cam_reg[WEIGHWIN_BASE + 4] = (unsigned char) (x2 >> 8);
cam_reg[WEIGHWIN_BASE + 6] = (unsigned char) x2;

cam_reg[WEIGHWIN_BASE + 8] = (unsigned char) (y1 >> 8);
cam_reg[WEIGHWIN_BASE + 10] = (unsigned char) y1;
cam_reg[WEIGHWIN_BASE + 12] = (unsigned char) (y2 >> 8);
cam_reg[WEIGHWIN_BASE + 14] = (unsigned char) y2;
return 0;
}

/*! Enable/Disable AWB and AE
 * \param awb 1 mean AWB enabled, 0 mean disabled
 * \param ae 1 mean AE enabled, 0 mean disabled
 * \sa Datasheet p. 50
 */

void e_po3030k_set_awb_ae(int awb, int ae) {
char val;
if(awb && ae) {
val = 0x3c;
} else {
if(awb) {
val = 0x2c;
} else {
if(ae) {
val = 0x1c;
} else {
val = 0x0c;
}
}
}
cam_reg[AWBAEENABLE_BASE] = val;
}

/*! Set flicker detection mode
 * \param manual Non-zero mean manual mode is enabled ( default automatic mode enabled )
 * \sa Datasheet p.29 e_po3030k_set_flicker_detection() e_po3030k_set_flicker_man_set()
 */

void e_po3030k_set_flicker_mode(int manual) {
if(manual) {
cam_reg[FLICKM_BASE] &= ~0x80;
} else {
cam_reg[FLICKM_BASE] |= 0x80;
}
}

/*! Set the 50/60Hz flicker detection
 * \param hz50 Non-zero mean 50Hz flicker detection enabled (default disabled)
 * \param 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.
 * \sa Datasheet p.29 e_po3030k_set_flicker_mode() e_po3030k_set_flicker_man_set()
 */

void e_po3030k_set_flicker_detection(int hz50, int hz60) {

if(hz50) {
cam_reg[FLICKM_BASE] &= ~0x40;
} else {
cam_reg[FLICKM_BASE] |= 0x40;
}

if(hz60) {
cam_reg[FLICKM_BASE] &= ~0x20;
} else {
cam_reg[FLICKM_BASE] |= 0x20;
}

}

/* Private internal function to get the real pixel clock */
static long po3030k_get_pixelclock(void) {
long clk;
switch (cam_reg[SPEED_ADDR]) {
case SPEED_2:
clk = MCLK/2;
break;
case SPEED_2_3:
clk = (MCLK*2)/3;
break;
case SPEED_4:
clk = MCLK/4;
break;
case SPEED_8:
clk = MCLK/8;
break;
case SPEED_16:
clk = MCLK/16;
break;
```

```
case SPEED_32:
clk = MCLK/32;
break;
case SPEED_64:
clk = MCLK/64;
break;
case SPEED_128:
clk = MCLK/128;
break;
default:
return 0;
}
if(cam_reg[COLOR_M_ADDR] == MODE_GRAYSCALE)
clk *= 2;
return clk;
}

/*! Set the camera's manual flicker's detection setting
 * \param hz50 The Hz for the 50Hz detection
 * \param hz60 The Hz for the 60Hz detection
 * \param fdm Flicker duration mode
 * \param fk Flicker count step
 * \param tol Flicker tolerance
 * \warning You must have set the mode ( image size, color ) before calling this function
 * \return Non-zero if an error occur, 0 if OK
 * \sa Datasheet p.29-30 e_po3030k_set_flicker_detection() e_po3030k_set_flicker_mode()
 */

int  e_po3030k_set_flicker_man_set(int hz50, int hz60, int fdm, int fk, int tol) {
unsigned int p50, p60;
/* checking if parameter are correct */
if(fk < 0 || fk > 3)
return -1;

if(tol < 0 || tol > 3)
return -1;

/* The datasheet isn't clear about this formula
 * At p.30 it's written:
 * Period = PixelClock / (HZ * 2 * FrameWidth)
 * But at p.60 it's written:
 * Period = 256*(MasterClock /(FrameWidth * 2))/(HZ*2)
   *        = 64*MasterClock / (FrameWidth * HZ)
 */

p50 = ( po3030k_get_pixelclock() * 2 * FRAME_WIDTH) / (long) hz50;
p60 = ( po3030k_get_pixelclock() * 2 * FRAME_WIDTH) / (long) hz60;

cam_reg[FLICKP_BASE] = (unsigned char) (p50 >> 8);
cam_reg[FLICKP_BASE + 2] = (unsigned char) p50;
cam_reg[FLICKP_BASE + 4] = (unsigned char) (p60 >> 8);
cam_reg[FLICKP_BASE + 6] = (unsigned char) p60;

if(fdm) {
cam_reg[FLICKM_BASE] |= 0x10;
} else {
cam_reg[FLICKM_BASE] &= ~0x10;
}

cam_reg[FLICKM_BASE] = (tol & 0x3) + ((fk & 0x3) << 2) + (cam_reg[FLICKM_BASE] & 0xF0);

return 0;
}
#endif
```

2.5.4 e_timers.c

```
/*! \file
 * \ingroup camera2
 * \brief Manage camera's interrupts (three timers)
 */

/* Compile this file with maximum optimisation
 * ( -O3 )
 */


#include <p30f6014a.h>
#include "./../../motor_LED/e_epuck_ports.h"
#include "e_po3030k.h"

/*! The number of row we should take
 * \sa e_po3030k_aApply_timer_config
 */
static int max_row;

/*! The number of bytes per row we should take
 * \sa e_po3030k_get_bytes_per_pixel, e_po3030k_apply_timer_config
 */
static int max_col;

/*! The buffer to write to */
static char * buffer;

/*! The flag to tell, the image is ready or not
 * Zero mean capture is in progress, non-zero mean capture done.
 * \sa e_po3030k_is_img_ready
 */
static int img_ready;

/*! The current row we are */
static int current_row;
/*! The current column we are */
static int current_col;
/*! Counter which is incremented each time we acquire a byte */
```

```
static int buf_pos;

/*! Number of bytes per pixel */
static int bytes_per_pixel;

/*! The current byte we are inside a pixel */
static int bpp_current;

/*! Pixel to "jump" between each effective pixel */
static int pixel_betw_pixel;

/*! The current pixel we are between two effective pixel */
static int pbp_current;

/*! The number of blank row between each effective row */
static int blank_betw_lines;

/*! The current row we ignore between each effective row */
static int bbl_current;



/*! \brief 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
 */
void __attribute__((interrupt, auto_psv))
_T4Interrupt(void) {
IFS1bits.T4IF = 0;

/* Yes, this is needed, otherwise we miss half of the interrupt
 * Hardware bug ? */
TMR4 = 0;
if(!bbl_current) {
if(++current_row > max_row) {
T1CONbits.TON = 0;
T4CONbits.TON = 0;
img_ready = 1;
} else {
/* We are in the correct column, let's enable pixel clock */
T1CONbits.TON = 1;
/* we must manually fire the first interrupt */
IFS0bits.T1IF = 1;
}
}
if(++bbl_current > blank_betw_lines)
bbl_current = 0;
}


/*! \brief 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 )
 */
void __attribute__((interrupt, auto_psv))
_T5Interrupt(void) {
IFS1bits.T5IF = 0;
/* let's enable Hsync */
T4CONbits.TON = 1;
/* single shot */
T5CONbits.TON = 0;
}


/*! \brief 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.
 */
void __attribute__((interrupt, auto_psv))
_T1Interrupt(void) {
int b;
int take_it;
static int p = 0;
IFS0bits.T1IF = 0;
b = CAM_DATA;

/* Yes, this is needed, otherwise we miss half of the interrupt
 * Hardware bug ? */
TMR1 = 0;


if(pbp_current)
take_it = 0;
else
take_it = 1;

if(++bpp_current == bytes_per_pixel) {
bpp_current = 0;
if(pbp_current++ >= pixel_betw_pixel) {
pbp_current = 0;
}
}

if(!take_it)
return;

buffer[buf_pos++] = b >> 8;
if(++current_col >= max_col) {
T1CONbits.TON = 0;
current_col = 0;
pbp_current = 0;
}
if(p == 0)  {
LED0 = 1;
p = 1;
} else {
LED0 = 0;
```

```
p = 0;
}
}

static void init_timer5(void) {
/* external pin, 1:1 prescaler */
T5CON = 0x2;
TMR5 = 0;
PR5 = 1;
IFS1bits.T5IF = 0;
IEC1bits.T5IE = 1;
T5CONbits.TON = 1;
}

static void init_timer4(void) {
T4CON = 0x2;
TMR4 = 0;
PR4 = 1;
IFS1bits.T4IF = 0;
T4CONbits.TON = 0;
IEC1bits.T4IE = 1;
}

static void init_timer1(void) {
/* external pin, 1:1 prescaler, sync */
T1CON = 0x2;
TMR1 = 0;
PR1 = 1;
IFS0bits.T1IF = 0;
T1CONbits.TON = 0;
T1CONbits.TSYNC = 1;
IEC0bits.T1IE = 1;

}

/*! Launch a capture in the \a buf buffer
 * \param buf The buffer to write to
 * \sa e_po3030k_config_cam and e_po3030k_is_img_ready
 */
void e_po3030k_launch_capture(char * buf) {
buf_pos = 0;
current_col = 0;
current_row = 0;
img_ready = 0;
bpp_current = 0;
pbp_current = 0;
bbl_current = 0;
buffer = buf;
init_timer4();
init_timer1();
/* Timer5 must ALWAY be initialized as the last one */
init_timer5();

}

/*! Modify the interrupt configuration
 * \warning This is an internal function, use \a e_po3030k_config_cam
 * \param pixel_row The number of row to take
 * \param pixel_col The number of pixel to take each \a pixel_row
 * \param bpp The number of byte per pixel
 * \param pbp The number of pixel to ignore between each pixel
 * \param bbl The number of row to ignore between each line
 * \sa e_po3030k_get_bytes_per_pixel and e_po3030k_config_cam
 */
void e_po3030k_apply_timer_config(int pixel_row, int pixel_col, int bpp, int pbp, int bbl) {
max_row = pixel_row;
max_col = pixel_col * bpp;
bytes_per_pixel = bpp;
pixel_betw_pixel = pbp;
blank_betw_lines = bbl;
}

/*! Check if the current capture is finished
 * \return Zero if the current capture is in progress, non-zero if the capture is done.
 * \sa e_po3030k_launch_capture
 */
int e_po3030k_is_img_ready(void) {
return img_ready;
}
```

2.6 codec

2.6.1 e_common.inc

```
; Data Size Unit constants
.equ    WORD,           2
.equ    CPLXWORD,       4
.equ    BYTE,           1

;Constants used to initialize the DCI module
.equ  FS, 7200 ; due to BCB quantization, only 7200 etc... value are possible
.equ  FSCKD, FS * 256                       ; frame clock rate
.equ  FCY, 7372800*2                        ; device instruction rate
.equ  BCG,  ( FCY / ( 2 * FSCKD ) ) - 1     ; equation for DCI clock rate

.equ NUMSAMPSTORED, 13204 ; number of sample stored in the soundsample.s file

;Constants useful to initialize Si3000 Codec
.equ  MINUS_ONE, 0x8000
.equ  MINUS_ONE_WITH_SECONDARY,  0x8001
```

```
.equ  PLUS_ONE, 0x7FFE
.equ  PLUS_ONE_WITH_SECONDARY, 0x7FFF

; Si3000 Register Address Summary (Table 13, Si30000-DS11)
; Read Control Address has bit 13 set
; Write Control Address has bit 13 clear

.equ  READ_CONTROL_1,           0x2100
.equ  WRITE_CONTROL_1,          0x0100

.equ  READ_CONTROL_2,           0x2200
.equ  WRITE_CONTROL_2,          0x0200

.equ  READ_PLL1_DIVIDE_N1,      0x2300
.equ  WRITE_PLL1_DIVIDE_N1,     0x0300

.equ  READ_PLL1_MULTIPLY_M1,    0x2400
.equ  WRITE_PLL1_MULTIPLY_M1,   0x0400

.equ  READ_RX_GAIN_CONTROL_1,   0x2500
.equ  WRITE_RX_GAIN_CONTROL_1,  0x0500

.equ  READ_ADC_VOLUME_CONTROL,  0x2600
.equ  WRITE_ADC_VOLUME_CONTROL, 0x0600

.equ  READ_DAC_VOLUME_CONTROL,  0x2700
.equ  WRITE_DAC_VOLUME_CONTROL, 0x0700

.equ  READ_Status_Report,       0x2800
.equ  WRITE_Status_Report,      0x0800

.equ  READ_ANALOG_ATTENUATION,  0x2900
.equ  WRITE_ANALOG_ATTENUATION, 0x0900
```

2.6.2 e_const_sound.s

A huge assembly file containing sounds.

2.6.3 e_init_codec_slave.s

```
;
; File Notes:
;
; At this point the DCI module has been initialized without enabling DCI
; interrupts.
; We will now initialize the codec by using the DCI module in a polling-based
; fashion (as opposed to an interrupt-based fashion).
;
; During Codec Initialization, the communication will work as follows:
; 1. We load TXBUF0 with a control word that has an LSB set that informs the codec
;    that a "secondary" frame will be sent in the next frame.
; 2. We load TXBUF1 with the required setting for a control register in the Si3000.
;    For e.g. a value 0x0118 sent to the codec will initiate a Write operation to
;    Control Register 1 in the Si3000 codec. The value written to the register
;    will be 0x18.
; 3. On the first Frame Sync pulse, the DCI will send the Codec, the contents of
;    the TXBUF0 register.
; 4. On the next frame-sync pulse, the DCI will send the Codec, the contents of
;    the TXBUF1 register. Since the Codec is expecting a "secondary frame" on
;    this pulse, it will read/write a control register based on the command.
; 5. Steps 3 and 4 are repeated for various control registers.
;
; AFTER initializing the Si3000 codec, we will disable the DCI module,
; modify certain transmit properties, enable DCI interrupts and the eventually
; re-enable the DCI module itself.
;
; The following features of the DCI module will be modified after setting up the
; Codec:
; 1. The DCI module will continue to transmit/receive only on TimeSlot1.
; 2. Each time slot will be 16-bits long.
; 3. A total of 16 time slots will be transmitted before a Frame Sync pulse is
;    generated.
; 4. All four available buffer registers (TXBUF0-4) are set up for usage.
; 5. The DCI module will generate frame sync pulses at 7200 Hz.
;



.include "p30f6014A.inc"
.include "e_common.inc"


.global _e_init_codec_slave


.section .text

_e_init_codec_slave:

        push    w1
        push    w0



        mov     #PLUS_ONE_WITH_SECONDARY, w0    ;Load TXBUF0 with a value that
        mov     w0, TXBUF0                      ;requests a secondary frame


        mov     #WRITE_PLL1_DIVIDE_N1, w0       ;Load TXBUF1 with a value that
        ior     #0x0000, w0                     ;will initiate a WRITE to PLL
        mov     w0, TXBUF1                      ;Divide Control Register 3
```

```
                                        ;in the Si3000

                                        ;Synchronize DCI with Si3000 to
                                        ;ensure first frame synce is
                                        ;aligned with TXBUF0.
                                        ;This is not really required when
                                        ;the DCI is Master and Codec is
                                        ;the Slave but is required when
                                        ;the Codec is the Master and DCI
                                        ;is the Slave.

        bclr    TRISF, #RF0
        nop
        bclr    PORTF, #RF0             ;Reset codec
        repeat  #400                    ;Transmit a minimum pulse
        nop                             ;width of 5uS
        bset    PORTF, #RF0             ;Release codec

        bset    DCICON1, #DCIEN         ;Enable DCI module

                                        ;This ensures the first primary
                                        ;and secondary frame use data
                                        ;loaded already into TXBUF0
                                        ;and TXBUF1

not_empty1:
        btss    DCISTAT, #TMPTY         ;Wait until TXBUF0 and TXBUF1
        bra     not_empty1              ;have been moved to their shadow
                                        ;registers for transmission

        mov     #PLUS_ONE_WITH_SECONDARY, w0    ;Load TXBUF0 with a value that
        mov     w0, TXBUF0              ;requests a secondary frame

        mov     #WRITE_PLL1_MULTIPLY_M1, w0     ;Load TXBUF1 with a value that
        ior     #0x0013, w0             ;will initiate a WRITE to PLL
        mov     w0, TXBUF1              ;Multiply Register 4

not_empty2:
        btss    DCISTAT, #TMPTY         ;Wait until TXBUF0 and TXBUF1
        bra     not_empty2              ;have been moved to their shadow
                                        ;registers for transmission


        mov     #PLUS_ONE_WITH_SECONDARY, w0    ;Load TXBUF0 with a value that
        mov     w0, TXBUF0              ;requests a secondary frame

        mov     #WRITE_CONTROL_1, w0            ;Load TXBUF1 with a value that
        bset    w0, #3                  ;will power up line driver and
        bset    w0, #4                  ;the speaker driver in Si3000
        mov     w0, TXBUF1              ;Control Register 1

not_empty3:
        btss    DCISTAT, #TMPTY         ;Wait until TXBUF0 and TXBUF1
        bra     not_empty3              ;have been moved to their shadow
                                        ;registers for transmission



        mov     #PLUS_ONE_WITH_SECONDARY, w0    ;Load TXBUF0 with a value that
        mov     w0, TXBUF0              ;requests a secondary frame

        mov     #WRITE_CONTROL_2, w0            ;Load TXBUF1 with a value that
        mov     w0, TXBUF1              ;will initialize Control Register 2
                                        ;to 0x00

not_empty4:
        btss    DCISTAT, #TMPTY         ;Wait until TXBUF0 and TXBUF1
        bra     not_empty4              ;have been moved to their shadow
                                        ;registers for transmission


        mov     #PLUS_ONE_WITH_SECONDARY, w0    ;Load TXBUF0 with a value that
        mov     w0, TXBUF0              ;requests a secondary frame

        mov     #WRITE_ADC_VOLUME_CONTROL, w0   ;Load TXBUF1 with a value that
        bset    w0, #1                  ;sets the line out active in the
        mov     w0, TXBUF1              ;Si3000's ADC Control Register 6

not_empty5:
        btss    DCISTAT, #TMPTY         ;Wait until TXBUF0 and TXBUF1
        bra     not_empty5              ;have been moved to their shadow
                                        ;registers for transmission


        mov     #PLUS_ONE_WITH_SECONDARY, w0    ;Load TXBUF0 with a value that
        mov     w0, TXBUF0              ;requests a secondary frame

        mov     #WRITE_DAC_VOLUME_CONTROL, w0   ;Load TXBUF1 with a value that
        ior     #0x005F, w0             ;sets TX PGA Gain to 0dB and
        mov     w0, TXBUF1              ;activates L/R speaker in the
                                        ;DAC Volume Control Register 7

not_empty6:
        btss    DCISTAT, #TMPTY         ;Wait until TXBUF0 and TXBUF1
        bra     not_empty6              ;have been moved to their shadow
                                        ;registers for transmission

;At this point there is no need to load TXBUF registers again, since all initialization
;commands have been either loaded or transmitted.

;Note that the TMPTY bit only gives a status of whether or not TXBUF registers
;are ready to accept data. This does not tell you if the transmissions themselves were
;completed.
;So, we need to test the SLOT<3:0> bits in the DCISTAT register.
;While testing we will ensure that TXBUF0 and TXBUF1 shadow registers have been
;fully transmitted on TimeSlot0 two times each. This will guarantee that initialization
;of the codec is complete
```

```
        mov     #0x0F00, w1                     ;Set up a mask for the SLOT bits
txbuf0_complete1:
        mov     DCISTAT, w0                     ;Read the DCISTAT register
        and     w0, w1, w0                      ;Logical-AND to extract SLOT bits
        cp0     w0                              ;Wait until SLOT bits reach 0b0000
        bra     nz, txbuf0_complete1            ;Did TXBUF0 get transmitted?

txbuf1_complete1:
        mov     DCISTAT, w0                     ;Read the DCISTAT register
        and     w0, w1, w0                      ;Logical-AND to extract SLOT bits
        cp0     w0                              ;Wait until SLOT bits reach non-zero
        bra     z, txbuf1_complete1             ;Did TXBUF1 get transmitted?

txbuf0_complete2:
        mov     DCISTAT, w0                     ;Read the DCISTAT register
        and     w0, w1, w0                      ;Logical-AND to extract SLOT bits
        cp0     w0                              ;Wait until SLOT bits reach 0b0000
        bra     nz, txbuf0_complete2            ;Did TXBUF0 get transmitted?

txbuf1_complete2:
        mov     DCISTAT, w0                     ;Read the DCISTAT register
        and     w0, w1, w0                      ;Logical-AND to extract SLOT bits
        cp0     w0                              ;Wait until SLOT bits reach 0b0000
        bra     z, txbuf1_complete2             ;Did TXBUF0 get transmitted?


        btss    DCISTAT, #ROV                   ;Testing overflow bit
        bra     disable_module                  ;No need to perform the dummy reads


        mov     RXBUF0, w0                      ;Dummy reads to clear
        mov     RXBUF1, w0                      ;overflow conditions

disable_module:
        bclr    DCICON1, #DCIEN                 ;Disable the DCI module

;Now re-initialize some aspects of the DCI communication for normal interrupt-driven
;operation.

        mov     #0b0000000111101111, w0         ;Data word size is 16-bits (bits 3:0)
        mov     w0, DCICON2                     ;data frame is 16 words (bits 8:5)


        bset    DCICON2, #BLEN1                 ;set buffer length control (4 data
        bset    DCICON2, #BLEN0                 ;words will be buffered between interrupts)


        bclr    IFS2, #DCIIF                    ;Clear the interrupt flag

        bset    IPC10, #DCIIP2                  ;Set Interrupt Priority to 4
        bclr    IPC10, #DCIIP1
        bclr    IPC10, #DCIIP0

        bset    IEC2, #DCIIE                    ;Enable DCI interrupts
        bset    DCICON1, #DCIEN                 ;Re-enable the DCI module

        pop     w0                              ;Restore w0, w1
        pop     w1

        return                                  ;Return to calling routine


.end                                            ;EOF
```

2.6.4 e_init_dci_master.s

```
;
; File Notes:
; 1. We will initialize the DCI module so as to allow it to initialize the
;    Codec in the codec initialization routine.
; 2. Some aspects of DCI functionality will be re-initialized after the Codec
;    setup is complete.
; 3. When this routine is executed, the DCI module will be set up for
;    transmitting and receiving only on TimeSlot1.
; 4. Each time slot will be 16-bits long.
;    A total of 8 time slots will be transmitted before a Frame Sync pulse is
;    generated.
; 5. Two of the four available buffer registers (TXBUF0/1) are set up for usage.



.include "p30f6014A.inc"
.include "e_common.inc"


.global  _e_init_dci_master

.section .text
_e_init_dci_master:

        push    w0                      ;Save w0


        clr     DCICON1                 ;Ensure DCI module is placed in known state
        clr     DCICON2
        clr     DCICON3
        clr     TSCON
        clr     RSCON

        bset    TRISB, #RB7             ;Make codec reset switch port pin an input for now.
                                        ;Ensure port pins are set correctly
                                        ;(Not really required since DCI takes
```

```
                                    ; control of it's pins)
        bclr    TRISG, #RG13        ;CSDO pin configured for output
        bset    TRISG, #RG12        ;CSDI pin configured for input
        bclr    TRISG, #RG15        ;COFS output to Si3000 in DCI Master mode
        bclr    TRISG, #RG14        ;CSCK output to Si3000

        bclr    DCICON1, #COFSM0    ;Frame sync mode (bits 1:0)
        bclr    DCICON1, #COFSM1    ;Set for multichannel frame sync mode
        bclr    DCICON1, #DJST      ;Data justification control
                                    ;Data txmit/rx begins 1 clock after Fs pulse
        bclr    DCICON1, #CSCKE     ;Sample clock edge control bit
                                    ;Data changes on rising, sampled on falling
        bset    DCICON1, #CSDOM     ;Tristate output CSDO pin when not txing
        bclr    DCICON1, #COFSD     ;Frame sync generated by dsPIC DCI
        bclr    DCICON1, #CSCKD     ;Clock is output from DCI


        mov     #0b0000000011101111, W0 ;Data word size is 16-bits (bits 3:0)
        mov     W0, DCICON2         ;Data frame is 8 words (bits 8:5)

        bclr    DCICON2, #BLEN1     ;Set buffer length control (2 data words
        bset    DCICON2, #BLEN0     ;Will be buffered between interrupts)


        mov     #BCG, W0            ;See common.inc file
        mov     W0, DCICON3         ;Initialize DCI bit clock generator


        bset    RSCON, #0           ;Set receive slot 1 enable
        bset    TSCON, #0           ;Set transmit slot 1 enable


        bclr    IFS2, #DCIIF        ;Ensure DCI interrupt flag is reset
        bclr    IEC2, #DCIIE        ;ISR processing is disabled for now
        bclr    DCICON1, #DCIEN     ;Ensure module is disabled for now

        pop     w0                  ;Restore w0

        return                      ;Return to calling routine


.end                                ;EOF
```

2.6.5 e_isr_dci.s

```
;
; File Notes:
;
; 1. The DCI interrupt service routine transfers 4 new tone samples to TXBUF0-4.
;    It also keeps a count of the number of samples transferred thus far.
;
; 2.  A tone sample is 100 milliseconds long and is followed by
;     15 milliseconds of silence.
;
; 3.  A tone sample is obtained by adding amplitudes of the sinusoid components.
;



.include "p30f6014A.inc"
.include "e_common.inc"


.global __DCIInterrupt
.section .text
__DCIInterrupt:
        bclr    IFS2, #DCIIF                    ;Clear the interrupt flag
        push.d  w0                              ;Save w0-w3 off to stack
        push.d  w2
        mov     RXBUF0, w0                      ;Dummy read of RXBUF0-RXBUF3
        mov     RXBUF1, w0
        mov     RXBUF2, w0
        mov     RXBUF3, w0
cp0 _e_stop_flag
bra     nz, spcl_exit_DCI_ISR
        cp0     e_samples_count                ;Is the number of DTMF samples
        bra     z, spcl_exit_DCI_ISR            ;samples have been SENT? If so, then exit

sendsamples:                                    ;If not then send next DTMF sample
        push    PSVPAG                          ;DTMF samples are constructed
        push    CORCON                          ;from sine samples stored in
        bset    CORCON, #PSV                    ;Program Memory, so use PSV to
        mov     e_actual_page, w0
        mov     w0, PSVPAG                      ;Set up PSVPAG register
        mov     e_tone_ptr, w1                  ;Retrieve pointer to lower tone
                                                ;component into w1
        call    test_page
        mov     [w1], w0                        ;Read low-tone sample
        bclr    w0, #0                          ;Clear the LS bit of the resultant
                                                ;This ensures that the dsPIC does
                                                ;not accidentally request a secondary
                                                ;frame transmission from the codec.
                                                ;Note that the Si3000 codec only provides
                                                ;an 84 dB dynamic range
        mov     w0, TXBUF0                      ;Write the result to TXBUF0
        call    test_page
        mov     [w1], w0                        ;Perform the same steps as above to load
        bclr    w0, #0
        mov     w0, TXBUF1
        call    test_page
        mov     [w1], w0
        bclr    w0, #0
        mov     w0, TXBUF2
```

```
        call    test_page
        mov     [w1], w0
        bclr    w0, #0
        mov     w0, TXBUF3

        dec2    e_samples_count                ;Decrement the number of DTMF samples left
        dec2    e_samples_count                ;to transmit by 4

        mov     w1, e_tone_ptr                  ;Update the Low and High tone pointers
        mov     PSVPAG,w1
mov w1,e_actual_page
        pop     CORCON                          ;Restore CORCON
        pop     PSVPAG                          ;Restore PSVPAG
exit_DCI_ISR:
        pop.d   w2                              ;Restore w0, w1, w2 and w3
        pop.d   w0
        retfie                                  ;Return from Interrupt


spcl_exit_DCI_ISR:
        clr     _e_dci_unavailable                  ;If silence has also been transmitted
setm _e_stop_flag
        pop.d   w2                              ;then clear the DCIUnavailable flag.
        pop.d   w0                              ;Restore w0-w3
        retfie                                  ;Return from Interrupt


test_page:
        add     w1, #2, w1
        bra     nc, return_testpage

        mov     PSVPAG, w1
        add     w1, #1, w1       ; change page
        mov     w1, PSVPAG
        mov     #0x8000, w1
return_testpage:
        return

.end                                            ;EOF
```

2.6.6 e_sound.{h,c}

```
/*****************************************************************************

Sound module
Version 1.0 August 2005 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup sound
 * \brief Package to play basics sounds on the e-puck's speaker.
 *
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

/*! \defgroup sound Sound
 *
 * \section intro_sec Introduction
 * The e-puck has got a speaker on his top. This package is made to take
 * advantage of it,
 * by playing little sample.
 *
 * \section organisation_sec Package organisation
 * 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.
 * There are three functions: \ref e_init_sound(void), \ref e_play_sound(int sound_offset, int sound_length)
 * and \ref void e_close_sound(void). When you want to play a sound YOU HAVE TO call \ref e_init_sound(void)
 * at first, but only the first time.
 * \n To play a sound call \ref e_play_sound(int sound_offset, int sound_length). 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 \ref e_play_sound(int sound_offset, int sound_length), 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 \ref void e_close_sound(void), you have to recall e\ref e_init_sound(void) to play
 * sound again.
 *
 * \section cmd_sec 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.
 * \code
 * #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");}
 *  }
 * }
 * \endcode
 *
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

#ifndef _SOUND
#define _SOUND


/* functions */
void e_init_sound(void);
void e_play_sound(int sound_offset, int sound_length);
void e_close_sound(void);


/* DCI */
void e_init_dci_master(void);
void e_init_codec_slave(void);
void e_sub_dci_kickoff(int,int);



#endif
```

```
/*****************************************************************************

Sound module
Version 1.0 August 2005 Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2006-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup sound
 * \brief Package to play basics sounds on the e-puck's speaker.
 * \n For more info look at this: \ref sound
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

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

/*! \brief 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).
 */
void e_init_sound(void)  // init sound module using si3000 codec
{
int i;
AUDIO_ON = 1;
for (i=0;i<10000;i++);
e_init_dci_master();
e_init_codec_slave();
}

/*! \brief Play a sound
 * \param sound_nbr the begining of the sound
 * \param sound_length the length of the sound
 * \sa sound
 */
```

```
void e_play_sound(int sound_nbr, int sound_length)
{
__asm__ volatile ("setm    _e_dci_unavailable"); // set DCI as used
__asm__ volatile ("clr    _e_stop_flag");


e_sub_dci_kickoff(sound_nbr, sound_length);
}

/*! \brief Disable the sound
 * After that you can't play sound anymore, if you want to, you
 * have to call e_init_sound
 */
void e_close_sound(void)
{
AUDIO_ON = 0;
}
```

2.6.7 e_sub_dci_kickoff.s

```
; File Notes:
;
; 1. This subroutine maps a number received by the UART,
;    to a DTMF tone and kicks off DCI transmissions.
;
; 2. All tone samples are stored in Program Memory, so we use PSV to access
;    this data.
;
; 3. A tone sample is 100 milliseconds long and is followed by
;    15 milliseconds of silence.
;
; 4. A tone sample is obtained by adding amplitudes of the sinusoid components.
;    The low frequency omponent has a signal strength 8 dB higher than the
;    high frequency component

.include "p30f6014A.inc"
.include "e_common.inc"


.section .ndata,data,near
.align WORD

.global _e_dci_unavailable, _e_stop_flag, e_tone_ptr
.global e_samples_count, e_tones_record, e_actual_page

.section .nbss,bss,near
.align 2
e_tone_ptr:             .space WORD

.section                .ndata
.align  WORD

_e_stop_flag: .hword 0x0000

_e_dci_unavailable:        .hword 0x0000                ;Flag that indicates the DCI
                                                        ;module is transmitting a tone
                                                        ;Flag is used to prevent one tone
                                                        ;"over-writing" another
e_samples_count:        .hword 0x0000                   ;Keeps a count of number of DTMF
                                                        ;samples left to transmit
e_actual_page: .hword 0x0000


.global _e_sub_dci_kickoff
.section .text

; sound offset in parameter 0, length in parameter 1
_e_sub_dci_kickoff:
        push    w1                              ; save W1 and W2 on the stack
        push    w2
        sl      w0,#1,w0
        bra     initiateDCI

exit_dci_kickoff:
        pop     w2
        pop     w1
        return


initiateDCI:
load_samp_cnt:
        mov     w1, e_samples_count ;set sumber of samples
        clr     w2
        mov     #psvoffset(e_sound_sample), w1   ;get datapointer to the sample
        add     w0, w1, w1                      ;add our start offset
        bra     nc, continue1                   ;if our offset don't makes an overfflow of PSV address -> continue1
        mov     #0x8000, w2                     ;an overflow occured in PSV address.
        add     w1, w2, w1                      ;add 0x8000 to our offset bacause PSV address begin at 0x8000
        mov     #1, w2 ;set W2 to 1 to mark that our sample begin in one page after the begining
                                                ;of "sound_samples"

continue1:
        push    CORCON
        push    PSVPAG                          ;Enable PSV to access tones
        bset    CORCON, #PSV
        mov     #psvpage(e_sound_sample), w0     ;get the page of our samples data
        add     w0, w2, w0                      ;if our sample begins at the nexte page, add 1
        mov     w0, PSVPAG
        nop

        call    test_page
        mov     [w1], w0
        bclr    w0, #0                          ;LS-bit cleared so that the codec does not
        mov     w0, TXBUF0                      ;assume a secondary frame request is being made
        call    test_page
```

```
        mov     [w1], w0
        bclr    w0, #0                          ;LS-bit cleared so that the codec does not
        mov     w0, TXBUF1                      ;assume a secondary frame request is being made
        call    test_page
        mov     [w1], w0
        bclr    w0, #0
        mov     w0, TXBUF2                      ;Load TXBUF0-TXBUF3
        call    test_page
        mov     [w1], w0
        bclr    w0, #0
        mov     w0, TXBUF3

        dec2    e_samples_count                ;Decrement samples count by 4 since
        dec2    e_samples_count                ;we sent 4 samples out
        mov     w1, e_tone_ptr                  ;Update the High and Low tone pointers

        mov     PSVPAG,w1
mov w1,e_actual_page

        pop     PSVPAG                          ;Restore CORCON, PSVPAG
        nop
        pop     CORCON
        bclr    IFS2, #DCIIF                    ;Clear the DCI Interrupt flag

        bra     exit_dci_kickoff

test_page:
        add     w1, #2, w1
        bra     nc, return_testpage

        mov     PSVPAG, w1
        add     w1, #1, w1                      ; change page
        mov     w1, PSVPAG
        mov     #0x8000, w1
return_testpage:
        return
.end
```

2.7 contrib/LIS_sensors_turret

2.7.1 e_devantech.{h,c}

```
/***************************************************************
*  Devantech sensor of e-puck *
* Version 1.0 7.2006 *
* Alessandro Ambuehl *
***************************************************************/

/*! \file
 * \ingroup LIS_SENSOR_TURRET
 * \brief Devantech sensor of e-puck
 * \author Jonathan Besuchet
 */

/*! \defgroup LIS_SENSOR_TURRET LIS sensor turret
 *
 * \section comment_sect Introduction
 * This module is made for an extention of the e-puck.
 * No documentation is available for now.
 */

#ifndef _devantech
#define _devantech

#include "./../../I2C/e_I2C_master_module.h"
#include "./../../motor_led/e_epuck_ports.h"

// Initialisation & disable
void e_init_devantech(void);
void e_disable_devantech(void);

// Commands to initiate a ranging
unsigned int e_get_dist_devantech(char device_add);
unsigned int e_get_delay_devantech(char device_add);

// Optional informations
char e_get_sr_devantech(char device_add); // Read the Software Revision
unsigned int e_get_light_devantech(char device_add); // Read the light sensor. return 0x80 if not avaiable

// Settings
void e_set_gain_devantech(char device_add, char gain);
void e_set_range_devantech(char device_add, char range);

// write command
void e_i2cd_write (char device_add, char reg, char value);

// Read commands
char e_i2cd_readb(char device_add, char reg); //read 1 byte
unsigned int e_i2cd_readw(char device_add, char reg); //read 1 word

#endif
```

```
/***************************************************************
*  Devantech sensor of e-puck *
* Version 1.0 7.2006 *
* Alessandro Ambuehl *
***************************************************************/
```

```
#include "e_devantech.h"
#include "./../../I2C/e_I2C_master_module.h"
#include "./../../I2C/e_I2C_protocol.h"


void e_init_devantech(void)
{
e_i2c_init(); // init I2C communication
e_i2c_enable(); // enable interrupts of I2C
}

void e_disable_devantech(void)
{
e_i2c_disable(); // disable I2C communication
}

// start ranging, wait & return the distance in cm
unsigned int e_get_dist_devantech(char device_add)
{
unsigned int dist;
unsigned char byte1, byte0;
unsigned char revision =255;
e_i2cp_write (device_add, 0, 81);  // start ranging comand - cm

while ((revision==255)) // Cecking for completion of ranging (according datasheet)
{
revision = e_i2cp_read(device_add, 0);
}
byte1=e_i2cp_read(device_add, 2);
byte0=e_i2cp_read(device_add, 3);
dist=byte1;
dist=(dist<<8)+byte0;
return dist;
}

// start ranging, wait & return the delay in uS
unsigned int e_get_delay_devantech(char device_add)
{
int delay;
char revision = 255;
e_i2cd_write (device_add, 0, 82); // start ranging comand - us
// Cecking for completion of ranging (according datasheet)
while ((revision=255))
{
revision = e_i2cd_readb(device_add, 0);
}

delay=e_i2cd_readw(device_add, 2);
return delay;
}

// returns the software revision of the sensor
char e_get_sr_devantech(char device_add)
{
char sr;
sr=e_i2cd_readb(device_add, 0);
return sr;
}

// returns the light intensity
// Warning: the light intensity is not updated with this command
unsigned int e_get_light_devantech(char device_add)
{
char light;
light=e_i2cp_read(device_add, 1);
return light;
}

// Set the analogue gain of the sensor
void e_set_gain_devantech(char device_add, char gain)
{
e_i2cd_write (device_add, 1, gain);
}

// Set the maximl range (distance)
void e_set_range_devantech(char device_add, char range)
{
e_i2cd_write (device_add, 2, range);
}

// Write 1 byte to register reg of devantec sensor
void e_i2cd_write (char device_add, char reg, char value)
{
e_i2c_start();
e_i2c_write(device_add);    // Writing the device (slave) address
e_i2c_write(reg);      // Device register address
e_i2c_write(value);       // Data to device
e_i2c_stop();             // Ending the communication
e_i2c_reset();
}

//read 1 byte from the register reg of devantec sensor
char e_i2cd_readb(char device_add, char reg)
{
char value;

e_i2c_start(); // start
e_i2c_write(device_add);    // Device address
e_i2c_write(reg);      // Register address

e_i2c_restart(); // 2nd start
e_i2c_write(device_add+1);    // Device address
  e_i2c_read(&value);     // read single byte
e_i2c_nack(); // only 1 byte is being read, so send nack
e_i2c_stop();              // end read cycle
e_i2c_reset();
```

```
    return value;
}

// sequantial random read of 2 bytes (word)
/*unsigned int e_i2cd_readw(char device_add, char reg)
{
char byte1, byte2;
unsigned int dist=0;

e_i2c_start(); // start
e_i2c_write(device_add);    // Device address
e_i2c_write(reg);      // Register address

e_i2c_restart(); // 2nd start
e_i2c_write(device_add+1);    // Device address
  e_i2c_read(&byte1);     // read byte 1
e_i2c_ack(); // send ack
e_i2c_read(&byte2);     // read byte 2
e_i2c_nack(); // end of word, so send nack
e_i2c_stop();              // end read cycle
e_i2c_reset();

dist=(unsigned int)byte1;
dist=dist<<8;
dist=dist+(unsigned int)byte2;
dist=byte1;
return dist;
}*/
```

2.7.2 e_sensext.{h,c}

```
// file sensext.h

#ifndef _SENSEXT
#define _SENSEXT

/* definitions */
//extension module i2c address
#define  I2C_ADDR_SENSEXT 0b10100010

// I2C Devantech Adress Definitions
#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);   // to be called before starting using the Sharp
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);

#endif
```

```
/*****************************************************************
*     LIS sensor extension module on e-puck   *
* Walter Karlen & Alessandro Ambuehl *
* Version 1.2 24.8.2007 WK *
*  separated sensext from sharp, new definitions *
* Removed from main loop
* Version 1.1 10.2006 AA *
* Initial release *
*****************************************************************/

/*attention: betwen the e-puck and the Sharp, an inverter buffer
inverts the control lines (LED 1-5 and Vin)*/

//#include "a_d/e_ad_conv.h"
//#include "motor_led/e_epuck_ports.h"
#include "e_sensext.h"
#include "e_sharp.h"
//#include "e_devantech.h"
//#include <a_d/advance_ad_scan/e_ad_conv.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>

static int sensext_wait=0;

void e_stop_sensext_wait(void)
{
sensext_wait=0;
}

void e_start_sensext_wait(void)
{
sensext_wait=1;
}

int e_get_sensext_wait(void)
{
return sensext_wait;
}

void e_init_sensext(void)
{
e_activate_agenda(e_stop_sensext_wait,0);
}
```

```
int e_sensext_process(int* sensext_param, unsigned int* sensext_value) //check pointer
{

unsigned char sensext_byteH, sensext_byteL;
// unsigned int sensext_dist=0, sensext_light=0;
unsigned char revision =255;

if(sensext_param[0]==-1) // Read the Devantech sensor
{
e_i2cp_enable();

// sensext_dist=e_get_dist_devantech(I2C_ADDR_SRF10);
e_i2cp_write (I2C_ADDR_SRF10, 0, 81);  // start ranging comand - cm
while ((revision==255)) // Cecking for completion of ranging (according datasheet)
{
revision = e_i2cp_read(I2C_ADDR_SRF10, 0);
}
sensext_byteH=e_i2cp_read(I2C_ADDR_SRF10, 2);
sensext_byteL=e_i2cp_read(I2C_ADDR_SRF10, 3);
//sensext_dist=sensext_byteH;
//sensext_dist=(sensext_dist<<8)+sensext_byteL;
sensext_value[0] = sensext_byteH<<8;
sensext_value[0] += sensext_byteL;
sensext_value[1] = (unsigned char)e_i2cp_read(I2C_ADDR_SRF10, 1);
//sensext_dist = sensext_byteH<<8;
// sensext_dist += sensext_byteL;
// sensext_light = (unsigned char)e_i2cp_read(I2C_ADDR_SRF10, 1);
//sensext_value[0] = sensext_dist;
//sensext_value[1] = sensext_light;
e_i2cp_disable();

return TRUE;

  // sprintf(buffer,"w,%u,%u\r\n", sensext_dist, sensext_light);
// uart_send_text(buffer);
}
else if((sensext_param[0]>=0)&&(sensext_param[0]<=31)) // read the analog sensors
{
SHARP_LED1 = !(sensext_param[0]&1); // mask the bit. all 5 leds are individually set to 0 or 1
SHARP_LED2 = !(sensext_param[0]&2);
SHARP_LED3 = !(sensext_param[0]&4);
SHARP_LED4 = !(sensext_param[0]&8);
SHARP_LED5 = !(sensext_param[0]&16);

if (sensext_param[1]!=sensext_param[0])
{ //reset sensor (5ms) and wait 30 msec for stability when sensor configuration changes //!!!WALTER!!!!!
e_sharp_off();
//SHARP_VIN = 1;
e_start_sensext_wait();
e_set_agenda_cycle(e_stop_sensext_wait, 50); // function to call after 5 ms
while(e_get_sensext_wait());
e_set_agenda_cycle(e_stop_sensext_wait, 0);
e_sharp_on();
//SHARP_VIN = 0;
e_start_sensext_wait();
e_set_agenda_cycle(e_stop_sensext_wait, 300); // function to call after 30 ms (25ms for sharp stability & 5ms for conversion)
while(e_get_sensext_wait());
e_set_agenda_cycle(e_stop_sensext_wait, 0);
}
sensext_value[0] = e_get_acc(0); // read the sharp output
// sprintf(buffer,"w,%d\r\n", sensext_dist);
// uart_send_text(buffer);
sensext_param[1]=sensext_param[0];
return TRUE;

} else if(sensext_param[0]==-2) // read the cmps03
{
e_i2cp_enable();


sensext_byteL=e_i2cp_read(I2C_ADDR_CMPS03, 3); //!!!WALTER!!!!!

sensext_byteH=e_i2cp_read(I2C_ADDR_CMPS03, 2); //!!!WALTER!!!!!
// sensext_value[0] =sensext_byteH;
// sensext_value[0] =(sensext_dist<<8)+sensext_byteL;
sensext_value[0] = sensext_byteH<<8;
sensext_value[0] += sensext_byteL; //1/10 //!!!WALTER!!!!!
sensext_value[1] = (unsigned char)e_i2cp_read(I2C_ADDR_CMPS03, 1); //360/255 //!!!WALTER!!!!!
e_i2cp_disable();
// sprintf(buffer,"w,%d,%d\r\n", sensext_dist, sensext_light); // compass bearing, word and byte
// uart_send_text(buffer);
return TRUE;
}
else
{
return FALSE;
}
}
```

2.7.3 e_sharp.{h,c}

```
// file sharp.h

#ifndef _SHARP
#define _SHARP

/* definitions */


#define SHARP 5 //AxeX
#define SHARP_LED1 _LATG6 //sel0
#define SHARP_LED2 _LATG7 //sel1
#define SHARP_LED3 _LATG8 //sel2
#define SHARP_LED4 _LATG9 //sel3
```

```
#define SHARP_LED5 _LATB6 //AxeY
#define SHARP_VIN  _LATB7 //AxeZ

//AxeX - pas de dir pour l'analogique
#define SHARP_LED1_DIR _TRISG6 //sel0
#define SHARP_LED2_DIR _TRISG7 //sel1
#define SHARP_LED3_DIR _TRISG8 //sel2
#define SHARP_LED4_DIR _TRISG9 //sel3
#define SHARP_LED5_DIR _TRISB6 //AxeY
#define SHARP_VIN_DIR  _TRISB7 //AxeZ



/* functions */

void e_init_sharp(void);   // to be called before starting using the Sharp
int e_get_dist_sharp(void);  // to get analog value of Sharp

// fontions pour les LED et pour Vin
void e_set_sharp_led(unsigned int sharp_led_number, unsigned int value); // set sharp_led_number (1-5) to value (0=off 1=on higher=inverse)
void e_sharp_led_clear(void);

//not working.always on. Hardware problem?
void e_sharp_on(void);
void e_sharp_off(void);


#endif
```

```
/*****************************************************************
*  Sharp sensor of e-puck *
* Alessandro Ambuehl & Walter Karlen (WK) *

* Version 1.2 23.8.2007   WK *
*  removed sensext  *
* Version 1.1 10.2006  AA *
* initial release
*****************************************************************/

/*attention: betwen the e-puck and the Sharp, an inverter buffer
inverts the control lines (LED 1-5 and Vin)*/

#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>

//#include <motor_led/advance_one_timer/e_agenda.h>
//#include <a_d/advance_ad_scan/e_ad_conv.h>


// initialisation du capteur et des modules lis
void e_init_sharp(void)
{

SHARP_LED1_DIR = OUTPUT_PIN;
SHARP_LED2_DIR = OUTPUT_PIN;
SHARP_LED3_DIR = OUTPUT_PIN;
SHARP_LED4_DIR = OUTPUT_PIN;
SHARP_LED5_DIR = OUTPUT_PIN;
SHARP_VIN_DIR =  OUTPUT_PIN;

e_sharp_led_clear();
e_sharp_off();

}


int e_get_dist_sharp()
{
int dist;
dist = e_get_acc(0); // read the sharp output
return dist;
}

//not used
void e_set_sharp_led(unsigned int sharp_led_number, unsigned int value)
// sharp_led_number between 1 and 5, value 0 (off) or 1 (on)
// if sharp_led_number other than 1-5, all leds are set to value
{
switch(sharp_led_number)
{
case 1:
{
if(value>1)
SHARP_LED1 = SHARP_LED1^1;
else
SHARP_LED1 = !value;
break;
}
case 2:
{
if(value>1)
SHARP_LED2 = SHARP_LED2^1;
else
SHARP_LED2 = !value;
break;
}
case 3:
{
if(value>1)
SHARP_LED3 = SHARP_LED3^1;
else
```

```
SHARP_LED3 = !value;
break;
}
case 4:
{
if(value>1)
SHARP_LED4 = SHARP_LED4^1;
else
SHARP_LED4 = !value;
break;
}
case 5:
{
if(value>1)
SHARP_LED5 = SHARP_LED5^1;
else
SHARP_LED5 = !value;
break;
}
default:
SHARP_LED1 = SHARP_LED2 = SHARP_LED3 = SHARP_LED4 = SHARP_LED5 = !value;
}
}

void e_sharp_led_clear(void)
{
//Inverter 1=^off  0=^on
SHARP_LED1 = 1;
SHARP_LED2 = 1;
SHARP_LED3 = 1;
SHARP_LED4 = 1;
SHARP_LED5 = 1;
}

void e_sharp_on(void)
{
SHARP_VIN = 0;
}
void e_sharp_off(void)
{
SHARP_VIN = 1;
}
```

2.8 contrib/SWIS_com_module

2.8.1 ComModule{h,c}

```
/************************************************************************
*  Titre     *
* Version 1.0 Date   *
* Author                  *
*                                  *
************************************************************************/

/*! \file
 * \ingroup ComModule
 * \brief Radio communication
 * \author Jonathan Besuchet
 */

/*! \defgroup ComModule Radio communication
 *
 * \section comment_sect Introduction
 * This module is made for an extention of the e-puck.
 * No documentation is available for now.
 */

#ifndef _COM_MODULE
#define _COM_MODULE


#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

#include "p30f6014A.h"

void InitComModule(unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue);
// OwnGroup : module will only receive packet of that group ID.
// hardwareAttenuator : COM_MODULE_HW_ATTENUATOR_25DB or COM_MODULE_HW_ATTENUATOR_0DB
// softAttenuator : value from 1 (full power) to 31 (-25dB)

int IsModulePlugged(); // return 0 if nothing connected, 1 if module is here.

void SetRadioEnabledState(unsigned char mode);// enable (1) or disable (0) the radio part of the module
void SetHardwareAttenuator(unsigned char AttenuatorMode); // COM_MODULE_HW_ATTENUATOR_25DB or COM_MODULE_HW_ATTENUATOR_0DB
void SetSoftwareAttenuator(unsigned char AttenuatorValue); // value from 1 (full power) to 31 (-25dB)
void SetOwnGroup(unsigned char GroupID); // module will only receive packet of that group ID.
void SetOwnAddress(unsigned int ownaddress); // current address of the module

unsigned char GetRadioEnabledState(); // renturn if radio in enabled or not;
unsigned char GetHardwareAttenuator(); // return COM_MODULE_HW_ATTENUATOR_25DB or COM_MODULE_HW_ATTENUATOR_0DB
unsigned char GetSoftwareAttenuator(); // return value from 1 (full power) to 31 (-25dB)
unsigned char GetOwnGroup(); // return module will only receive packet of that group ID.

unsigned char GetStatus(); // return status register
//bit 0 : PACKET_READY_FLAG : a packet is in receive buffer. auto clear when last byte of receive buffer is read
//bit 1 : TX_IDLE_FLAG  : module is ready to send a new packet
```

```
//bit 2 : PACKET_LOST_FLAG  : a packet was in receive buffer when another packet arrived. 2.nd packet is lost. auto clear when last byte of receive buffer is read
//bit 3 : TX_SEND_ERROR : error durint packet send. auto clear during next send attempt.

int SendPacket(unsigned char destinationgroup, unsigned int destinationaddress, unsigned char* packet, int packetsize); // return 1 if OK, 0 if error
// destinationGroup : the destination group of the packet 0xFF is broadcast group
// destinationaddress : the address of the destination module. 0xff is broadcast address
// packet : unsigned char array of data to be sent
// packetSize : number of fields of the unsigned char array to be sent. ( max is COM_MODULE_MAXSIZE bytes )

int IsPacketReady(unsigned char* packet, int* packetSize); // return 1 and the packet + size if new packet receive. 0 if not.
// packet : unsigned char array of data to put receive data in. must be COM_MODULE_MAXSIZE bytes array.
// packetSize : the effective data size receive.

#endif
```

```
#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"

#define COM_MODULE_I2C_ADDR (0x3F<<1)

#define BUFFER_DATA_LENGTH  (COM_MODULE_MAXSIZE + 14)


#define STATUS_REG_ADDR    0x00  // status like PACKET_LOST_FLAG,TX_IDLE,PACKET_READY
#define CONFIG_REG_ADDR    0x01  // config like ..... hardware att. sleep...
#define SEND_REG_ADDR      0x02  // <> 0 -> send sendbuffer
#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 // AM_OSCOPE

#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 // how many byte after packet start should I write data ?

//status reg
#define PACKET_READY_FLAG 0x01 // bit 1
#define TX_IDLE_FLAG 0x02 // bit 2
#define PACKET_LOST_FLAG 0x04 // bit 3
#define TX_SEND_ERROR 0x08 // bit 4

//config reg
#define HARDWAREATT_SET_FLAG 0x01 //bit 1
#define RADIO_ENABLED_FLAG 0x80 //bit 8

// send reg
#define REQUEST_TO_SEND_FLAG 0x01

unsigned char ReadRegister(unsigned char registeraddr) {
return (0x00FF & e_i2cp_read(COM_MODULE_I2C_ADDR, registeraddr));
}

void WriteRegister(unsigned char registeraddr, unsigned char value) {
e_i2cp_write(COM_MODULE_I2C_ADDR, registeraddr, value);
}

void InitComModule(unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue) {
e_i2cp_init();
e_i2cp_enable();

while (IsModulePlugged() == 0); // wait till module online

SetOwnGroup(owngroup);
SetOwnAddress(ownaddress);
SetHardwareAttenuator(hardwareattenuatormode);
SetSoftwareAttenuator(softwareattenuatorvalue);
SetRadioEnabledState(1);
}

int IsModulePlugged() {
unsigned char value;
value = GetStatus();
if (value == 0xFF) // value read are always 0xff when module not plugged in
return 0;
return 1;
}

void SetRadioEnabledState(unsigned char mode) {
unsigned char old_value;
old_value = ReadRegister(CONFIG_REG_ADDR);
if (mode != 0)
WriteRegister(CONFIG_REG_ADDR, old_value |= RADIO_ENABLED_FLAG);
else
WriteRegister(CONFIG_REG_ADDR, old_value &= ~RADIO_ENABLED_FLAG);
}

void SetHardwareAttenuator(unsigned char attenuatormode) {
unsigned char old_value;
old_value = ReadRegister(CONFIG_REG_ADDR);
if (attenuatormode != 0)
```

```
WriteRegister(CONFIG_REG_ADDR, old_value |= HARDWAREATT_SET_FLAG);
else
WriteRegister(CONFIG_REG_ADDR, old_value &= ~HARDWAREATT_SET_FLAG);
}


void SetSoftwareAttenuator(unsigned char attenuatorvalue) {
if (attenuatorvalue > 31)
attenuatorvalue = 31;
WriteRegister(SOFTATT_REG_ADDR, attenuatorvalue);
}

void SetOwnGroup(unsigned char owngroup) {
WriteRegister(OWNGROUP_REG_ADDR, owngroup);
}

void SetOwnAddress(unsigned int ownaddress) {
WriteRegister(OWNADDRL_REG_ADDR, ownaddress & 0xFF);
ownaddress=ownaddress>>8;
WriteRegister(OWNADDRH_REG_ADDR, ownaddress & 0xFF);
}

unsigned char GetHardwareAttenuator() {
if (ReadRegister(CONFIG_REG_ADDR) & HARDWAREATT_SET_FLAG);
return 1;
return 0;
}

unsigned char GetRadioEnabledState() {
if (ReadRegister(CONFIG_REG_ADDR) & RADIO_ENABLED_FLAG);
return 1;
return 0;
}

unsigned char GetSoftwareAttenuator() {
return ReadRegister(SOFTATT_REG_ADDR);
}

unsigned char GetOwnGroup() {
return ReadRegister(OWNGROUP_REG_ADDR);
}

unsigned int GetOwnAddress() {
unsigned char lowbyte, highbyte;
lowbyte=ReadRegister(OWNADDRL_REG_ADDR);
highbyte=ReadRegister(OWNADDRH_REG_ADDR);
return lowbyte | ((int)highbyte<<8);
}

unsigned char GetStatus() {
return ReadRegister(STATUS_REG_ADDR);
}

int SendPacket(unsigned char destinationgroup, unsigned int destinationaddress, unsigned char* packet, int packetsize) {
unsigned int i;
if (packetsize > COM_MODULE_MAXSIZE)
return 0; // not allowed -> return error

while ((GetStatus() & TX_IDLE_FLAG) == 0); // wait till module is ready to transmit

WriteRegister(SEND_BUFFER_START + AM_MSGTYPE_IN_PACKET_OFFSET, AM_MSGTYPE);
WriteRegister(SEND_BUFFER_START + GROUPDATA_IN_PACKET_OFFSET, destinationgroup);
WriteRegister(SEND_BUFFER_START + ADDRLDATA_IN_PACKET_OFFSET, destinationaddress & 0xFF);
destinationaddress=destinationaddress>>8;
WriteRegister(SEND_BUFFER_START + ADDRHDATA_IN_PACKET_OFFSET, destinationaddress & 0xFF);
for (i=0; i<packetsize; i++) {
WriteRegister(FIRSTDATA_IN_PACKET_OFFSET+SEND_BUFFER_START+i, packet[i]);
}
WriteRegister(SEND_REG_ADDR,1); // send packet
// maybe wait here
if (GetStatus() & TX_SEND_ERROR) // flag TX error set -> return error
return 0;
return 1; // return OK
}

int IsPacketReady(unsigned char* packet, int* packetsize) {
unsigned int i;
if ((packet == NULL) || (packetsize == NULL))
return 0; // return withour reading NULL pointer means uC reset if write
if (GetStatus() & PACKET_READY_FLAG) { // a packet is ready !
*packetsize = ReadRegister(REC_BUFFER_START); // get size of received buffer
if (*packetsize > COM_MODULE_MAXSIZE)
return 0; // error in transmission. size is too big and impossible
for (i = 0; i< *packetsize ; i++) {
packet[i] = ReadRegister(REC_BUFFER_START+FIRSTDATA_IN_PACKET_OFFSET+i);
}
ReadRegister(REC_BUFFER_END);
return 1;
}
for(i=0; i<0x1000; i++); // wait
return 0; // no packet ready -> 0
}
```

2.9 fft

2.9.1 e_fast_copy.s

```
; Author : Florian Vaussard
; Last modified : 22.08.2006

.text
.global _e_fast_copy
_e_fast_copy:
```

```
; W0 : input array
; W1 : output array
; W2 : arrays size

DEC W2, W2
MOV #0, W5
DO W2,end
MOV [W0++],[W1++] ; copy the element in the real part
end: MOV W5, [W1++] ; put a 0 in the imag part

RETURN

.end
```

2.9.2 e_fft_utilities.h

```
/************************************************************************************

Fast Fourier transform module
August 2007: first version Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2007 Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************************/

/*! \file
 * \ingroup fft
 * \brief Some fft features.
 *
 * Here you have two functions that are really usefull to work
 * with the FFT
 * \author Code & doc: Jonathan Besuchet
 */
#ifndef _FFT_UTILITIES
#define _FFT_UTILITIES

/*! \brief Copy an array to an other array, using REPEAT instruction
 * \param in_array The array from which you want to copy
 * \param out_array The destination array
 * \param size The number of element you want to copy
 */
void e_fast_copy(int* in_array, int* out_array, int size);

/*! \brief Substract the mean from the samples of the array (-> produce zero-mean samples)
 * \param array The array that you want to produce zero-mean samples
 * \param size The size of the array
 * \param log2size The log in base 2 of the size
 */
void e_subtract_mean(int* array, int size, int log2size);

#endif
```

2.9.3 e_fft.{h,c}

```
/************************************************************************************

Fast Fourier transform module
August 2007: first version Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2007 Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************************/

/*! \file
 * \ingroup fft
 * \brief Package to mange the FFT.
 *
 * In this package the FFT (fast fourier transform) is done with
 * the special dsPic 30fxxxx instructions.
 * \n Before calling the e_doFFT_asm() 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
 */

/*! \defgroup fft FFT
 *
 * \section intro_sec 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.
 * \n The dsPic has some specials instructions (MAC instructions)
 * which are used here.
 *
```

```
 * \section howitwork_sec 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 \ref FFT_BLOCK_LENGTH (it's in the
 *   file e_fft.h).
 *
 * 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) function
 * - to call the e_doFFT_asm(fractcomplex* sigCmpx) function
 *
 * A little code to illustrate this.
 * \code
 *  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);
 * \endcode
 *
 * \author Doc: Jonathan Besuchet
 */

#ifndef FFT_H
#define FFT_H

#include <dsp.h>

/* Constant Definitions */

// Number of frequency points in the FFT */
#define FFT_BLOCK_LENGTH 256

// Number of "Butterfly" Stages in FFT processing
#if (FFT_BLOCK_LENGTH == 64)
#define LOG2_BLOCK_LENGTH  6
#endif
#if (FFT_BLOCK_LENGTH == 128)
#define LOG2_BLOCK_LENGTH  7
#endif
#if (FFT_BLOCK_LENGTH == 256)
#define LOG2_BLOCK_LENGTH  8
#endif
#if (FFT_BLOCK_LENGTH == 512)
#define LOG2_BLOCK_LENGTH  9
#endif

void e_doFFT_asm(fractcomplex* sigCmpx);

#endif // FFT_H
```

```
/************************************************************************************

Fast Fourier transform module
August 2007: first version Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2007 Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************************/

/*! \file
 * \ingroup fft
 * \brief Package to mange the FFT.
 * \author Code & doc: Jonathan Besuchet
 */

#include <dsp.h>

#include "e_fft.h"

/* Twiddle Factor array in Program memory */
extern const fractcomplex twiddleFactors[FFT_BLOCK_LENGTH/2] __attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2)));

/*! \brief Execute the FFT with dsPic special instructions
 * \param sigCmpx The pointer of the begining of the array on
 * which you want to perform the FFT.
 */
void e_doFFT_asm(fractcomplex* sigCmpx){
/* Perform FFT operation */
FFTComplexIP (LOG2_BLOCK_LENGTH, &sigCmpx[0], (fractcomplex *) __builtin_psvoffset(&twiddleFactors[0]), (int) __builtin_psvpage(&twiddleFactors[0]));

/* Store output samples are stored in bit-reversed order of their addresses. -> Reorder them */
BitReverseComplex (LOG2_BLOCK_LENGTH, &sigCmpx[0]);
}
```

2.9.4 e_input_signal.c

```
/************************************************************************

Fast Fourier transform module
August 2007: first version Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2007 Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************/

/*! \file
 * \ingroup fft
 * \brief 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.
 * \n \n This array has the following size: 64, 128, 256, 512 depending
 * the choice you have made in fft.h
 * \author Code & doc: Jonathan Besuchet
 */

#include <dsp.h>
#include "e_fft.h"

#if (FFT_BLOCK_LENGTH == 64)
fractcomplex sigCmpx[FFT_BLOCK_LENGTH]  __attribute__ ((section (".ydata, data, ymemory"), aligned (FFT_BLOCK_LENGTH * 2 *2))) =
{
0,9514,5857,-5909,-9495,62,9533,5806,
-5959,-9475,125,9552,5755,-6010,-9455,188,
9570,5703,-6060,-9435,251,9588,5651,-6110,
-9413,314,9606,5599,-6160,-9392,377,9623,
5547,-6209,-9370,440,9640,5495,-6258,-9348,
502,9657,5442,-6307,-9326,565,9673,5389,
-6356,-9303,628,9689,5336,-6404,-9279,691,
9704,5283,-6452,-9256,754,9719,5229,-6500
};
#endif
#if (FFT_BLOCK_LENGTH == 128)
fractcomplex sigCmpx[FFT_BLOCK_LENGTH]  __attribute__ ((section (".ydata, data, ymemory"), aligned (FFT_BLOCK_LENGTH * 2 *2))) =
{
0,9514,5857,-5909,-9495,62,9533,5806,
-5959,-9475,125,9552,5755,-6010,-9455,188,
9570,5703,-6060,-9435,251,9588,5651,-6110,
-9413,314,9606,5599,-6160,-9392,377,9623,
5547,-6209,-9370,440,9640,5495,-6258,-9348,
502,9657,5442,-6307,-9326,565,9673,5389,
-6356,-9303,628,9689,5336,-6404,-9279,691,
9704,5283,-6452,-9256,754,9719,5229,-6500,
-9232,816,9734,5175,-6548,-9207,879,9748,
5122,-6595,-9183,942,9762,5067,-6643,-9158,
1004,9775,5013,-6689,-9132,1067,9788,4959,
-6736,-9106,1129,9801,4904,-6782,-9080,1192,
9813,4849,-6829,-9054,1254,9825,4794,-6874,
-9027,1316,9836,4739,-6920,-8999,1379,9848,
4683,-6965,-8972,1441,9858,4627,-7010,-8944
};
#endif
#if (FFT_BLOCK_LENGTH == 256)
fractcomplex sigCmpx[FFT_BLOCK_LENGTH]  __attribute__ ((section (".ydata, data, ymemory"), aligned (FFT_BLOCK_LENGTH * 2 *2))) =
{
0,9514,5857,-5909,-9495,62,9533,5806,
-5959,-9475,125,9552,5755,-6010,-9455,188,
9570,5703,-6060,-9435,251,9588,5651,-6110,
-9413,314,9606,5599,-6160,-9392,377,9623,
5547,-6209,-9370,440,9640,5495,-6258,-9348,
502,9657,5442,-6307,-9326,565,9673,5389,
-6356,-9303,628,9689,5336,-6404,-9279,691,
9704,5283,-6452,-9256,754,9719,5229,-6500,
-9232,816,9734,5175,-6548,-9207,879,9748,
5122,-6595,-9183,942,9762,5067,-6643,-9158,
1004,9775,5013,-6689,-9132,1067,9788,4959,
-6736,-9106,1129,9801,4904,-6782,-9080,1192,
9813,4849,-6829,-9054,1254,9825,4794,-6874,
-9027,1316,9836,4739,-6920,-8999,1379,9848,
4683,-6965,-8972,1441,9858,4627,-7010,-8944,
1503,9869,4572,-7055,-8916,1565,9879,4516,
-7099,-8887,1627,9888,4459,-7143,-8858,1690,
9897,4403,-7187,-8828,1751,9906,4346,-7231,
-8799,1813,9914,4290,-7274,-8769,1875,9922,
4233,-7317,-8738,1937,9930,4176,-7360,-8708,
1999,9937,4119,-7402,-8676,2060,9944,4061,
-7445,-8645,2122,9950,4004,-7486,-8613,2183,
9956,3946,-7528,-8581,2244,9962,3888,-7569,
-8549,2306,9967,3830,-7610,-8516,2367,9972,
3772,-7651,-8483,2428,9977,3713,-7691,-8449,
2489,9981,3655,-7731,-8415,2550,9984,3596,
-7771,-8381,2610,9988,3538,-7810,-8347,2671,
9990,3479,-7850,-8312,2732,9993,3420,-7888,
-8277,2792,9995,3361,-7927,-8241,2852,9997,
3301,-7965,-8206,2913,9998,3242,-8003,-8169,
2973,9999,3182,-8040,-8133,3033,9999,3123,
-8078,-8096,3093,9999,3063,-8115,-8059,3152,
9999,3003,-8151,-8022,3212,9998,2943,-8188,
-7984,3272,9997,2883,-8223,-7946,3331,9996,
2822,-8259,-7908,3390,9994,2762,-8294,-7869,
3449,9992,2701,-8329,-7830,3508,9989,2641,
-8364,-7791,3567,9986,2580,-8398,-7751,3626,
```

```
9983,2519,-8432,-7711,3684,9979,2458,-8466,
-7671,3743,9974,2397,-8499,-7631,3801,9970,
2336,-8532,-7590,3859,9965,2275,-8565,-7549,
3917,9959,2214,-8597,-7507,3975,9954,2152,
-8629,-7466,4032,9947,2091,-8661,-7424,4090,
9941,2029,-8692,-7381,4147,9934,1968,-8723,
-7339,4204,9926,1906,-8754,-7296,4261,9919,
1844,-8784,-7253,4318,9910,1782,-8814,-7209,
4375,9902,1720,-8843,-7165,4431,9893,1659,
-8872,-7121,4487,9883,1596,-8901,-7077,4544,
9874,1534,-8930,-7033,4600,9864,1472,-8958,
-6988,4655,9853,1410,-8986,-6943,4711,9842,
1348,-9013,-6897,4766,9831,1285,-9040,-6851,
4821,9819,1223,-9067,-6806,4876,9807,1160,
-9093,-6759,4931,9794,1098,-9119,-6713,4986,
9782,1035,-9145,-6666,5040,9768,973,-9170,
-6619,5095,9755,910,-9195,-6572,5149,9741,
848,-9220,-6524,5202,9726,785,-9244,-6476,
5256,9712,722,-9268,-6428,5309,9696,659,
-9291,-6380,5363,9681,597,-9314,-6332,5416,
9665,534,-9337,-6283,5468,9649,471,-9359,
-6234,5521,9632,408,-9381,-6184,5573,9615,
345,-9403,-6135,5625,9597,282,-9424,-6085,
5677,9579,220,-9445,-6035,5729,9561,157,
-9465,-5985,5780,9543,94,-9485,-5934,5831,
9524,31,-9505,-5883,5882,9504,-32,-9525,
-5832,5933,9484,-95,-9544,-5781,5984,9464
};
#endif
#if (FFT_BLOCK_LENGTH == 512)
fractcomplex sigCmpx[FFT_BLOCK_LENGTH]  __attribute__ ((section (".ydata, data, ymemory"), aligned (FFT_BLOCK_LENGTH * 2 *2))) =
{
0,9514,5857,-5909,-9495,62,9533,5806,
-5959,-9475,125,9552,5755,-6010,-9455,188,
9570,5703,-6060,-9435,251,9588,5651,-6110,
-9413,314,9606,5599,-6160,-9392,377,9623,
5547,-6209,-9370,440,9640,5495,-6258,-9348,
502,9657,5442,-6307,-9326,565,9673,5389,
-6356,-9303,628,9689,5336,-6404,-9279,691,
9704,5283,-6452,-9256,754,9719,5229,-6500,
-9232,816,9734,5175,-6548,-9207,879,9748,
5122,-6595,-9183,942,9762,5067,-6643,-9158,
1004,9775,5013,-6689,-9132,1067,9788,4959,
-6736,-9106,1129,9801,4904,-6782,-9080,1192,
9813,4849,-6829,-9054,1254,9825,4794,-6874,
-9027,1316,9836,4739,-6920,-8999,1379,9848,
4683,-6965,-8972,1441,9858,4627,-7010,-8944,
1503,9869,4572,-7055,-8916,1565,9879,4516,
-7099,-8887,1627,9888,4459,-7143,-8858,1690,
9897,4403,-7187,-8828,1751,9906,4346,-7231,
-8799,1813,9914,4290,-7274,-8769,1875,9922,
4233,-7317,-8738,1937,9930,4176,-7360,-8708,
1999,9937,4119,-7402,-8676,2060,9944,4061,
-7445,-8645,2122,9950,4004,-7486,-8613,2183,
9956,3946,-7528,-8581,2244,9962,3888,-7569,
-8549,2306,9967,3830,-7610,-8516,2367,9972,
3772,-7651,-8483,2428,9977,3713,-7691,-8449,
2489,9981,3655,-7731,-8415,2550,9984,3596,
-7771,-8381,2610,9988,3538,-7810,-8347,2671,
9990,3479,-7850,-8312,2732,9993,3420,-7888,
-8277,2792,9995,3361,-7927,-8241,2852,9997,
3301,-7965,-8206,2913,9998,3242,-8003,-8169,
2973,9999,3182,-8040,-8133,3033,9999,3123,
-8078,-8096,3093,9999,3063,-8115,-8059,3152,
9999,3003,-8151,-8022,3212,9998,2943,-8188,
-7984,3272,9997,2883,-8223,-7946,3331,9996,
2822,-8259,-7908,3390,9994,2762,-8294,-7869,
3449,9992,2701,-8329,-7830,3508,9989,2641,
-8364,-7791,3567,9986,2580,-8398,-7751,3626,
9983,2519,-8432,-7711,3684,9979,2458,-8466,
-7671,3743,9974,2397,-8499,-7631,3801,9970,
2336,-8532,-7590,3859,9965,2275,-8565,-7549,
3917,9959,2214,-8597,-7507,3975,9954,2152,
-8629,-7466,4032,9947,2091,-8661,-7424,4090,
9941,2029,-8692,-7381,4147,9934,1968,-8723,
-7339,4204,9926,1906,-8754,-7296,4261,9919,
1844,-8784,-7253,4318,9910,1782,-8814,-7209,
4375,9902,1720,-8843,-7165,4431,9893,1659,
-8872,-7121,4487,9883,1596,-8901,-7077,4544,
9874,1534,-8930,-7033,4600,9864,1472,-8958,
-6988,4655,9853,1410,-8986,-6943,4711,9842,
1348,-9013,-6897,4766,9831,1285,-9040,-6851,
4821,9819,1223,-9067,-6806,4876,9807,1160,
-9093,-6759,4931,9794,1098,-9119,-6713,4986,
9782,1035,-9145,-6666,5040,9768,973,-9170,
-6619,5095,9755,910,-9195,-6572,5149,9741,
848,-9220,-6524,5202,9726,785,-9244,-6476,
5256,9712,722,-9268,-6428,5309,9696,659,
-9291,-6380,5363,9681,597,-9314,-6332,5416,
9665,534,-9337,-6283,5468,9649,471,-9359,
-6234,5521,9632,408,-9381,-6184,5573,9615,
345,-9403,-6135,5625,9597,282,-9424,-6085,
5677,9579,220,-9445,-6035,5729,9561,157,
-9465,-5985,5780,9543,94,-9485,-5934,5831,
9524,31,-9505,-5883,5882,9504,-32,-9525,
-5832,5933,9484,-95,-9544,-5781,5984,9464,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
};
#endif
```

2.9.5 e_subtract_mean.s

```
; Author : Florian Vaussard
; Last modified : 22.08.2006

.text

.global _e_subtract_mean
_e_subtract_mean:
; W0 : input array
; W1 : array size
; W2 : log2(array size)

; Save the address of the array
PUSH W0

CLR A
; Set up REPEAT loop with count 'ARRAYSIZE' (passed in W1)
DEC W1, W1
REPEAT W1
; Add all the elements of the array in the A acc.
ADD [W0++], A

; Faut encore dcaler de 16 pour retrouver le bon rsultat
; Dcalage par log2(size)
SFTAC A, W2

; Result of the mean
MOV ACCAH, W3

; Retrieve the address of the array
POP W0

; Save the address of the array
PUSH W0
DO W1, loop1
; Subtract the mean from the array
MOV [W0], W4
loop1: SUB W4, W3, [W0++]

; Retrieve the address of the array
POP W0
RETURN

.end
```

2.9.6 e_twiddle_factors.c

```
/*****************************************************************
*  2005 Microchip Technology Inc.
*
* FileName:        twiddleFactors.c
* Dependencies:    Header (.h) files if applicable, see below
* Processor:       dsPIC30Fxxxx
* Compiler:        MPLAB C30 v1.33.00 or higher
* IDE:             MPLAB IDE v7.20.01 or later
* Dev. Board Used: dsPICDEM 1.1 Development Board
* Hardware Dependencies: None
*
* SOFTWARE LICENSE AGREEMENT:
* Microchip Technology Inc. (Microchip) licenses this software to you
* solely for use with Microchip dsPIC digital signal controller
* products. The software is owned by Microchip and is protected under
* applicable copyright laws.  All rights reserved.
*
* SOFTWARE IS PROVIDED AS IS.  MICROCHIP EXPRESSLY DISCLAIMS ANY
* WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL MICROCHIP
* BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL
* DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF
* PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS
* BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),
* ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
*
* REVISION HISTORY:
*-----------------------------------------------------------------------
* Author            Date      Comments on this revision
*-----------------------------------------------------------------------
* HV                09/30/05  First release of source file
*
*-----------------------------------------------------------------------
*
* ADDITIONAL NOTES:
*
*
*****************************************************************/

/*! \file
 * \ingroup fft
 * \brief The FFT factor from Microchip
 * \author Jonathan Besuchet
 */

#include <dsp.h>
#include "e_fft.h"

#if (FFT_BLOCK_LENGTH == 64)
        const fractcomplex twiddleFactors[] __attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2)))=
        {
        0x7FFF, 0x0000, 0x7F62, 0xF374, 0x7D8A, 0xE707, 0x7A7D, 0xDAD8,
        0x7642, 0xCF04, 0x70E3, 0xC3A9, 0x6A6E, 0xB8E3, 0x62F2, 0xAECC,
        0x5A82, 0xA57E, 0x5134, 0x9D0E, 0x471D, 0x9592, 0x3C57, 0x8F1D,
        0x30FC, 0x89BE, 0x2528, 0x8583, 0x18F9, 0x8276, 0x0C8C, 0x809E,
        0x0000, 0x8000, 0xF374, 0x809E, 0xE707, 0x8276, 0xDAD8, 0x8583,
        0xCF04, 0x89BE, 0xC3A9, 0x8F1D, 0xB8E3, 0x9592, 0xAECC, 0x9D0E,
        0xA57D, 0xA57D, 0x9D0E, 0xAECC, 0x9592, 0xB8E3, 0x8F1D, 0xC3A9,
        0x89BE, 0xCF04, 0x8583, 0xDAD8, 0x8276, 0xE707, 0x809E, 0xF374
        } ;
#endif
#if (FFT_BLOCK_LENGTH == 128)
        const fractcomplex twiddleFactors[] __attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2)))=
        {
        0x7FFF, 0x0000, 0x7FD9, 0xF9B8, 0x7F62, 0xF374, 0x7E9D, 0xED38,
        0x7D8A, 0xE707, 0x7C2A, 0xE0E6, 0x7A7D, 0xDAD8, 0x7885, 0xD4E1,
        0x7642, 0xCF04, 0x73B6, 0xC946, 0x70E3, 0xC3A9, 0x6DCA, 0xBE32,
        0x6A6E, 0xB8E3, 0x66D0, 0xB3C0, 0x62F2, 0xAECC, 0x5ED7, 0xAA0A,
        0x5A82, 0xA57E, 0x55F6, 0xA129, 0x5134, 0x9D0E, 0x4C40, 0x9930,
        0x471D, 0x9592, 0x41CE, 0x9236, 0x3C57, 0x8F1D, 0x36BA, 0x8C4A,
        0x30FC, 0x89BE, 0x2B1F, 0x877B, 0x2528, 0x8583, 0x1F1A, 0x83D6,
        0x18F9, 0x8276, 0x12C8, 0x8163, 0x0C8C, 0x809E, 0x0648, 0x8027,
        0x0000, 0x8000, 0xF9B8, 0x8027, 0xF374, 0x809E, 0xED38, 0x8163,
        0xE707, 0x8276, 0xE0E6, 0x83D6, 0xDAD8, 0x8583, 0xD4E1, 0x877C,
        0xCF04, 0x89BE, 0xC946, 0x8C4A, 0xC3A9, 0x8F1D, 0xBE32, 0x9236,
        0xB8E3, 0x9592, 0xB3C0, 0x9931, 0xAECC, 0x9D0E, 0xAA0A, 0xA129,
        0xA57E, 0xA57E, 0xA129, 0xAA0A, 0x9D0E, 0xAECC, 0x9931, 0xB3C0,
        0x9592, 0xB8E3, 0x9236, 0xBE32, 0x8F1D, 0xC3A9, 0x8C4A, 0xC946,
        0x89BE, 0xCF04, 0x877C, 0xD4E1, 0x8583, 0xDAD8, 0x83D6, 0xE0E6,
        0x8276, 0xE707, 0x8163, 0xED38, 0x809E, 0xF374, 0x8027, 0xF9B8
        } ;
#endif
#if (FFT_BLOCK_LENGTH == 256)
        const fractcomplex twiddleFactors[] __attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2))) =
        {
0x7FFF,0x0000,0x7FF6,0xFCDC,0x7FD9,0xF9B8,0x7FA7,0xF695,
0x7F62,0xF374,0x7F0A,0xF055,0x7E9D,0xED38,0x7E1E,0xEA1E,
0x7D8A,0xE707,0x7CE4,0xE3F4,0x7C2A,0xE0E6,0x7B5D,0xDDDC,
0x7A7D,0xDAD8,0x798A,0xD7D9,0x7884,0xD4E1,0x776C,0xD1EF,
0x7642,0xCF04,0x7505,0xCC21,0x73B6,0xC946,0x7255,0xC673,
0x70E3,0xC3A9,0x6F5F,0xC0E9,0x6DCA,0xBE32,0x6C24,0xBB85,
0x6A6E,0xB8E3,0x68A7,0xB64C,0x66CF,0xB3C0,0x64E8,0xB140,
0x62F2,0xAECC,0x60EC,0xAC65,0x5ED7,0xAA0A,0x5CB4,0xA7BD,
0x5A82,0xA57E,0x5843,0xA34C,0x55F6,0xA129,0x539B,0x9F14,
0x5134,0x9D0E,0x4EC0,0x9B18,0x4C40,0x9931,0x49B4,0x9759,
0x471D,0x9592,0x447B,0x93DC,0x41CE,0x9236,0x3F17,0x90A1,
0x3C57,0x8F1D,0x398D,0x8DAB,0x36BA,0x8C4A,0x33DF,0x8AFB,
0x30FC,0x89BE,0x2E11,0x8894,0x2B1F,0x877C,0x2827,0x8676,
0x2528,0x8583,0x2224,0x84A3,0x1F1A,0x83D6,0x1C0B,0x831C,
0x18F9,0x8276,0x15E2,0x81E3,0x12C8,0x8163,0x0FAB,0x80F7,
0x0C8C,0x809E,0x096B,0x8059,0x0648,0x8028,0x0324,0x800A,
0x0000,0x8000,0xFCDC,0x800A,0xF9B8,0x8028,0xF695,0x8059,
0xF374,0x809E,0xF055,0x80F7,0xED38,0x8163,0xEA1E,0x81E3,
0xE707,0x8276,0xE3F5,0x831C,0xE0E6,0x83D6,0xDDDC,0x84A3,
```

```
0xDAD8,0x8583,0xD7D9,0x8676,0xD4E1,0x877C,0xD1EF,0x8894,
0xCF04,0x89BE,0xCC21,0x8AFB,0xC946,0x8C4A,0xC673,0x8DAB,
0xC3A9,0x8F1D,0xC0E9,0x90A1,0xBE32,0x9236,0xBB85,0x93DC,
0xB8E3,0x9593,0xB64C,0x975A,0xB3C0,0x9931,0xB140,0x9B18,
0xAECC,0x9D0E,0xAC65,0x9F14,0xAA0A,0xA129,0xA7BD,0xA34C,
0xA57E,0xA57E,0xA34C,0xA7BD,0xA129,0xAA0A,0x9F14,0xAC65,
0x9D0E,0xAECC,0x9B18,0xB140,0x9931,0xB3C0,0x975A,0xB64C,
0x9593,0xB8E3,0x93DC,0xBB85,0x9236,0xBE32,0x90A1,0xC0E9,
0x8F1D,0xC3A9,0x8DAB,0xC673,0x8C4A,0xC946,0x8AFB,0xCC21,
0x89BF,0xCF04,0x8894,0xD1EF,0x877C,0xD4E1,0x8676,0xD7D9,
0x8583,0xDAD8,0x84A3,0xDDDC,0x83D6,0xE0E6,0x831C,0xE3F5,
0x8276,0xE707,0x81E3,0xEA1E,0x8163,0xED38,0x80F7,0xF055,
0x809E,0xF374,0x8059,0xF695,0x8028,0xF9B8,0x800A,0xFCDC,

        } ;
#endif
#if (FFT_BLOCK_LENGTH == 512 )
        const fractcomplex twiddleFactors[] __attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2))) =
        {
        0x7FFF, 0x0000, 0x7FFE, 0xFE6E, 0x7FF6, 0xFCDC, 0x7FEA, 0xFB4A,
        0x7FD9, 0xF9B8, 0x7FC2, 0xF827, 0x7FA7, 0xF695, 0x7F87, 0xF505,
        0x7F62, 0xF374, 0x7F38, 0xF1E4, 0x7F0A, 0xF055, 0x7ED6, 0xEEC6,
        0x7E9D, 0xED38, 0x7E60, 0xEBAB, 0x7E1E, 0xEA1E, 0x7DD6, 0xE892,
        0x7D8A, 0xE707, 0x7D3A, 0xE57D, 0x7CE4, 0xE3F4, 0x7C89, 0xE26D,
        0x7C2A, 0xE0E6, 0x7BC6, 0xDF61, 0x7B5D, 0xDDDC, 0x7AEF, 0xDC59,
        0x7A7D, 0xDAD8, 0x7A06, 0xD958, 0x798A, 0xD7D9, 0x790A, 0xD65C,
        0x7885, 0xD4E1, 0x77FB, 0xD367, 0x776C, 0xD1EF, 0x76D9, 0xD079,
        0x7642, 0xCF04, 0x75A6, 0xCD92, 0x7505, 0xCC21, 0x7460, 0xCAB2,
        0x73B6, 0xC946, 0x7308, 0xC7DB, 0x7255, 0xC673, 0x719E, 0xC50D,
        0x70E3, 0xC3A9, 0x7023, 0xC248, 0x6F5F, 0xC0E9, 0x6E97, 0xBF8C,
        0x6DCA, 0xBE32, 0x6CF9, 0xBCDA, 0x6C24, 0xBB85, 0x6B4B, 0xBA33,
        0x6A6E, 0xB8E3, 0x698C, 0xB796, 0x68A7, 0xB64C, 0x67BD, 0xB505,
        0x66D0, 0xB3C0, 0x65DE, 0xB27F, 0x64E9, 0xB140, 0x63EF, 0xB005,
        0x62F2, 0xAECC, 0x61F1, 0xAD97, 0x60EC, 0xAC65, 0x5FE4, 0xAB36,
        0x5ED8, 0xAA0A, 0x5DC8, 0xA8E2, 0x5CB4, 0xA7BD, 0x5B9D, 0xA69C,
        0x5A83, 0xA57E, 0x5964, 0xA463, 0x5843, 0xA34C, 0x571E, 0xA238,
        0x55F6, 0xA128, 0x54CA, 0xA01C, 0x539B, 0x9F14, 0x5269, 0x9E0F,
        0x5134, 0x9D0E, 0x4FFB, 0x9C11, 0x4EC0, 0x9B17, 0x4D81, 0x9A22,
        0x4C40, 0x9930, 0x4AFB, 0x9843, 0x49B4, 0x9759, 0x486A, 0x9674,
        0x471D, 0x9592, 0x45CD, 0x94B5, 0x447B, 0x93DC, 0x4326, 0x9307,
        0x41CE, 0x9236, 0x4074, 0x9169, 0x3F17, 0x90A1, 0x3DB8, 0x8FDD,
        0x3C57, 0x8F1D, 0x3AF3, 0x8E62, 0x398D, 0x8DAB, 0x3825, 0x8CF8,
        0x36BA, 0x8C4A, 0x354E, 0x8BA0, 0x33DF, 0x8AFB, 0x326E, 0x8A5A,
        0x30FC, 0x89BE, 0x2F87, 0x8927, 0x2E11, 0x8894, 0x2C99, 0x8805,
        0x2B1F, 0x877B, 0x29A4, 0x86F6, 0x2827, 0x8676, 0x26A8, 0x85FA,
        0x2528, 0x8583, 0x23A7, 0x8511, 0x2224, 0x84A3, 0x209F, 0x843A,
        0x1F1A, 0x83D6, 0x1D93, 0x8377, 0x1C0C, 0x831C, 0x1A83, 0x82C6,
        0x18F9, 0x8276, 0x176E, 0x822A, 0x15E2, 0x81E2, 0x1455, 0x81A0,
        0x12C8, 0x8163, 0x113A, 0x812A, 0x0FAB, 0x80F6, 0x0E1C, 0x80C8,
        0x0C8C, 0x809E, 0x0AFB, 0x8079, 0x096B, 0x8059, 0x07D9, 0x803E,
        0x0648, 0x8027, 0x04B6, 0x8016, 0x0324, 0x800A, 0x0192, 0x8002,
        0x0000, 0x8000, 0xFE6E, 0x8002, 0xFCDC, 0x800A, 0xFB4A, 0x8016,
        0xF9B8, 0x8027, 0xF827, 0x803E, 0xF695, 0x8059, 0xF505, 0x8079,
        0xF374, 0x809E, 0xF1E4, 0x80C8, 0xF055, 0x80F6, 0xEEC6, 0x812A,
        0xED38, 0x8163, 0xEBAB, 0x81A0, 0xEA1E, 0x81E2, 0xE892, 0x822A,
        0xE707, 0x8276, 0xE57D, 0x82C6, 0xE3F4, 0x831C, 0xE26D, 0x8377,
        0xE0E6, 0x83D6, 0xDF61, 0x843A, 0xDDDC, 0x84A3, 0xDC59, 0x8511,
        0xDAD8, 0x8583, 0xD958, 0x85FA, 0xD7D9, 0x8676, 0xD65C, 0x86F6,
        0xD4E1, 0x877B, 0xD367, 0x8805, 0xD1EF, 0x8894, 0xD079, 0x8927,
        0xCF04, 0x89BE, 0xCD92, 0x8A5A, 0xCC21, 0x8AFB, 0xCAB2, 0x8BA0,
        0xC946, 0x8C4A, 0xC7DB, 0x8CF8, 0xC673, 0x8DAB, 0xC50D, 0x8E62,
        0xC3A9, 0x8F1D, 0xC248, 0x8FDD, 0xC0E9, 0x90A1, 0xBF8C, 0x9169,
        0xBE32, 0x9236, 0xBCDA, 0x9307, 0xBB85, 0x93DC, 0xBA33, 0x94B5,
        0xB8E3, 0x9592, 0xB796, 0x9674, 0xB64C, 0x9759, 0xB505, 0x9843,
        0xB3C0, 0x9930, 0xB27F, 0x9A22, 0xB140, 0x9B17, 0xB005, 0x9C11,
        0xAECC, 0x9D0E, 0xAD97, 0x9E0F, 0xAC65, 0x9F14, 0xAB36, 0xA01C,
        0xAA0A, 0xA128, 0xA8E2, 0xA238, 0xA7BD, 0xA34C, 0xA69C, 0xA463,
        0xA57D, 0xA57D, 0xA463, 0xA69C, 0xA34C, 0xA7BD, 0xA238, 0xA8E2,
        0xA128, 0xAA0A, 0xA01C, 0xAB36, 0x9F14, 0xAC65, 0x9E0F, 0xAD97,
        0x9D0E, 0xAECC, 0x9C11, 0xB005, 0x9B17, 0xB140, 0x9A22, 0xB27F,
        0x9930, 0xB3C0, 0x9843, 0xB504, 0x9759, 0xB64C, 0x9674, 0xB796,
        0x9592, 0xB8E3, 0x94B5, 0xBA33, 0x93DC, 0xBB85, 0x9307, 0xBCDA,
        0x9236, 0xBE32, 0x9169, 0xBF8C, 0x90A1, 0xC0E9, 0x8FDD, 0xC248,
        0x8F1D, 0xC3A9, 0x8E62, 0xC50D, 0x8DAB, 0xC673, 0x8CF8, 0xC7DB,
        0x8C4A, 0xC946, 0x8BA0, 0xCAB2, 0x8AFB, 0xCC21, 0x8A5A, 0xCD92,
        0x89BE, 0xCF04, 0x8927, 0xD079, 0x8894, 0xD1EF, 0x8805, 0xD367,
        0x877B, 0xD4E1, 0x86F6, 0xD65C, 0x8676, 0xD7D9, 0x85FA, 0xD958,
        0x8583, 0xDAD8, 0x8510, 0xDC59, 0x84A3, 0xDDDC, 0x843A, 0xDF61,
        0x83D6, 0xE0E6, 0x8377, 0xE26D, 0x831C, 0xE3F4, 0x82C6, 0xE57D,
        0x8275, 0xE707, 0x8229, 0xE892, 0x81E2, 0xEA1E, 0x81A0, 0xEBAB,
        0x8163, 0xED38, 0x812A, 0xEEC6, 0x80F6, 0xF055, 0x80C8, 0xF1E4,
        0x809E, 0xF374, 0x8079, 0xF505, 0x8059, 0xF695, 0x803E, 0xF827,
        0x8027, 0xF9B8, 0x8016, 0xFB4A, 0x800A, 0xFCDC, 0x8002, 0xFE6E
        } ;
#endif
```

2.9.7 e_libdsp-coff.a

A .a file.

2.9.8 e_libdsp-elf.a

A .a file.

2.10 I2C

2.10.1 e_I2C_master_module.{h,c}

```
/************************************************************************

I2C master module
Version 1.0 may 2005 Davis Daidie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Davis Daidie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

************************************************************************/

/*! \file
 * \ingroup i2c
 * \brief Manage I2C basics.
 *
 * \author Code: Davis Daidie \n Doc: Jonathan Besuchet
 */

/*! \defgroup i2c I2C
 *
 * \section intro_sec Introduction
 * This software allows using the I2C hardware module on a DsPic30f60xx
 * in a master mode for a single master system.
 * \n 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 (\ref e_i2c_init(void))
 * - sending the Start bit (\ref e_i2c_start(void))
 * - sending the Restart bit (\ref e_i2c_restart(void))
 * - sending the Stop bit (\ref e_i2c_stop(void))
 * - sending the acknowledgement bit (\ref e_i2c_ack(void))
 * - writing or receiving a byte (\ref e_i2c_write(char byte),
 *  \ref e_i2c_read(char *buf))
 * - ...
 *
 * \section prot_sect 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.
 * \subsection prot_subsect1 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.
 * \n 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:
 * \n \n 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.
 *
 * \subsection prot_subsect2 Start and Stop conditions
 * Within the procedure of the I2C-bus, unique situations
 * arise which are defined as START (S) and STOP (P)
 * conditions.
 * \n A HIGH to LOW transition on the SDA line while SCL is
 * HIGH is one such unique case. This situation indicates a
 * START condition.
 * \n A LOW to HIGH transition on the SDA line while SCL is
 * HIGH defines a STOP condition.
 * \n 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.
 * \n 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.
 *
 * \subsection prot_subsect3 Transfering data
 * Every byte put on the SDA line must be 8-bits long (char type).
```

```
 * \n \n Acknowledge:
 * \n 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.
 * \n 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.
 * \n Usually, a receiver which has been addressed is obliged to
 * generate an acknowledge after each byte has been received.
 * \n 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.
 * \n 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.
 *
 * \section use_on_epuck_sect Use I2C on e-puck
 * On the e-puck, the microcontroller is always the master.
 * \subsection use_on_epuck_subsect1 Basics functions
 * The functions of the files e_I2C_master_module.c and
 * e_I2C_master_module.h are low level I2C functions.
 * \n They are made to perform the basics tasks like:
 * - initializing the I2C on the microcontroller (\ref e_i2c_init(void))
 * - sending the Start bit (\ref e_i2c_start(void))
 * - sending the Restart bit (\ref e_i2c_restart(void))
 * - sending the Stop bit (\ref e_i2c_stop(void))
 * - sending the acknowledgement bit (\ref e_i2c_ack(void))
 * - writing or receiving a byte (\ref e_i2c_write(char byte),
 *  \ref e_i2c_read(char *buf))
 * - ...
 * \subsection use_on_epuck_subsect2 More developed functions
 * The functions of the files e_I2C_protocol.c and
 * e_I2C_protocol.h are made to directly send or
 * receive data from or to a specified slave.
 *
 * \section reference 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
 */

#ifndef _I2C_MASTER_MODULE
#define _I2C_MASTER_MODULE

#include "p30f6014A.h"


#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

// -use I2C_init() in your initialisation
// -I2C_enable(void) before anythig else to enable interrupts
// the others functions must be use generate a valide i2c message.

// all the functions return 1 to confirme the oparation and 0 for an error
// in this case, use char I2C_reset(void) and redo the wrong oparation.
// if there is no result, switch of evrything and beginn from the beginning.. :-)

char e_i2c_init(void);
char e_i2c_start(void);
char e_i2c_restart(void);
char e_i2c_ack(void);
char e_i2c_nack(void);
char e_i2c_read(char *buf);
char e_i2c_stop(void);
char e_i2c_write(char byte);
char e_i2c_enable(void);
char e_i2c_disable(void);
char e_i2c_reset(void);


#endif
```

```
/*****************************************************************************

I2C master module
Version 1.0 may 2005 Davis Daidie
```

```
This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Davis Daidie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup i2c
 * \brief Manage I2C basics.
 *
 * This module manage the I2C basics functions (low level
 * I2C functions).
 * \n 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 \n Doc: Jonathan Besuchet
 */

#include "e_I2C_master_module.h"

char e_i2c_mode;
int  e_interrupts[3];

/*! \brief Wait until I2C Bus is Inactive */
void idle_i2c(void)
{
    while(I2CCONbits.SEN || I2CCONbits.PEN || I2CCONbits.RCEN || I2CCONbits.ACKEN || I2CSTATbits.TRSTAT);
}

/*! \brief Initialize the microcontroller for I2C uses
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_init(void)
{
long i;
I2CBRG=150; // frequency of SCL at 100kHz
I2CCONbits.I2CEN=0; // disable I2C
I2CCONbits.I2CEN=1; // enable I2C
IFS0bits.MI2CIF=0; // clear master interrupt flag
IFS0bits.SI2CIF=0; // clear slave interrupt flag
IPC3bits.MI2CIP=5; // priority level
// IEC0bits.MI2CIE=1; // enabhle master I2C interrupt
IEC0bits.SI2CIE=0; // diseble slave I2C interrupt


for(i=10000;i;i--);
return 1;
}

/*! \brief Reset the microcontroller for I2C uses
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_reset(void)
{
long i=0;
I2CCONbits.I2CEN=0; // disable I2C and stop peripheric
IFS0bits.MI2CIF=0; //
IFS0bits.SI2CIF=0; //
IEC0bits.SI2CIE=0; //
for(i=10000;i;i--);

e_i2c_init(); // intit I2C
e_i2c_enable(); //enable interrupt

for(i=10000;i;i--);

return 1;
}

/*! \brief Enable special I2C interrupt
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_enable(void)
{
/* e_interrupts[0]=IEC0;
IEC0=0;
e_interrupts[1]=IEC1;
IEC1=0;
e_interrupts[2]=IEC2;
IEC2=0; */

IFS0bits.MI2CIF=0; // clear master interrupt flag
IEC0bits.MI2CIE=1; // enable master I2C interrupt
return 1;
}

/*! \brief Disable special I2C interrupt
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_disable(void)
{

/* IEC0=e_interrupts[0];
IEC1=e_interrupts[1];
IEC2=e_interrupts[2];*/
```

```
IFS0bits.MI2CIF=0; // clear master interrupt flag
IEC0bits.MI2CIE=0; // disable master I2C interrupt
return 1;
}

/*! \brief Make the start bit
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_start(void)
{
long i;
e_i2c_mode=START;
if(I2CSTATbits.P)
{
I2CCONbits.SEN=1;
for(i=10000;i;i--)
if(!e_i2c_mode)
return 1;
return 0;
}else
return 0;
}

/*! \brief Make the restart bit
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_restart(void)
{
long i;
e_i2c_mode=RESTART;
if(I2CSTATbits.S)
{
I2CCONbits.RSEN=1;
for(i=10000;i;i--)
if(!e_i2c_mode)
return 1;
return 0;
}else
return 0;
}

/*! \brief Make the stop bit
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_stop(void)
{
long i;
e_i2c_mode=STOP;

I2CCONbits.PEN=1;

for(i=10000;i;i--)
if(!e_i2c_mode)
return 1;
return 0;
}

/*! \brief Make the acknowledgement bit
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_ack(void)
{
long i;
e_i2c_mode=ACKNOWLEDGE;

// make sure I2C bus is inactive
    if(I2CCONbits.SEN || I2CCONbits.PEN || I2CCONbits.RCEN || I2CCONbits.ACKEN || I2CCONbits.RSEN)
return 0;

// set ACK mode
I2CCONbits.ACKDT=0;
I2CCONbits.ACKEN=1;

for(i=10000;i;i--)
if(!e_i2c_mode)
return 1;
return 0;
}

/*! \brief Make the non-acknowledgement bit
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_nack(void)
{
long i;
e_i2c_mode=ACKNOWLEDGE;

// make sure I2C bus is inactive
    if(I2CCONbits.SEN || I2CCONbits.PEN || I2CCONbits.RCEN || I2CCONbits.ACKEN || I2CCONbits.RSEN)
return 0;

// set NACK mode
I2CCONbits.ACKDT=1;
I2CCONbits.ACKEN=1;

for(i=10000;i;i--)
if(!e_i2c_mode)
return 1;
return 0;
}

/*! \brief Read the I2C input register
 * \param buf A pointer to store the datas received
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_read(char *buf)
{
```

```
long i=10000;
char read_ok=0;
// int test=0;
e_i2c_mode=READ;

for(i=10000;i;i--)
if(!(I2CCONbits.SEN || I2CCONbits.PEN || I2CCONbits.RCEN || I2CCONbits.ACKEN || I2CSTATbits.TRSTAT))
{
read_ok=1;
break;
}
if(!read_ok)
return 0;

// start receive mode for I2C
I2CCONbits.RCEN=1;

// keep polling for I2C interrupt
for(i=100000;i;i--)
if(!e_i2c_mode) // once I2C interrupt is tripped, read buffer and return 1
{
// test=I2CSTAT; // used for debug purposes
*buf=I2CRCV;
return 1;
}
return 0;

}

/*! \brief Write on the I2C output register
 * \param byte What you want to send on I2C
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2c_write(char byte)
{
long i;
e_i2c_mode=WRITE;
I2CTRN=byte;

// poll for I2C interrupt
for(i=10000;i;i--)
if(!e_i2c_mode) // return 1(transmisison OK) if interrupt was tripped)
return 1;
return 0;
}

// interrupt  routine:
void  __attribute__((__interrupt__, auto_psv)) _MI2CInterrupt(void)
{
IFS0bits.MI2CIF=0; // clear master interrupt flag
e_i2c_mode=OPERATION_OK;
}
```

2.10.2 e_I2C_protocol.{h,c}

```
/*****************************************************************************

I2C master module
Version 1.0 may 2005 Davis Daidie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Davis Daidie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup i2c
 * \brief 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 \n Doc: Jonathan Besuchet
 */

#ifndef _I2C_PROTOCOL
#define _I2C_PROTOCOL

#include "e_I2C_master_module.h"
#include "../motor_led/e_epuck_ports.h"


//public interface

// use void e_i2cp_init(void); in your initialisation. no interrupt is enable
// use void e_i2cp_enable(void); before any other operation to enable the interrupts
// use e_i2cp_write,I2Cp_read to write or read in the Camera registers

void e_i2cp_init(void);
char e_i2cp_write (char device_add,char reg, char value);
char e_i2cp_read(char device_add,char reg);
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);
void e_i2cp_disable(void);
```

```
#endif
```

```
/*****************************************************************************

I2C master module
Version 1.0 may 2005 Davis Daidie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Davis Daidie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup i2c
 * \brief 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 \n Doc: Jonathan Besuchet
 */

#include "e_I2C_protocol.h"

/*! \brief Initialize the microcontroller for I2C uses
 * \return 1 to confirme the oparation and 0 for an error
 */
void e_i2cp_init(void)
{
e_i2c_init();
}

/*! \brief Enable special I2C interrupt
 * \return 1 to confirme the oparation and 0 for an error
 */
void e_i2cp_enable(void)
{
e_i2c_enable();
}

/*! \brief Disable special I2C interrupt
 * \return 1 to confirme the oparation and 0 for an error
 */
void e_i2cp_disable(void)
{
e_i2c_disable();
}

/*! \brief Read a specific register on a device
 * \param device_add The address of the device you want information
 * \param reg The register address you want read on the device
 * \return The readed value
 */
char e_i2cp_read(char device_add, char reg)
{
char error=0;
char value;
while(!error)
{
error=1;
error&=e_i2c_start();
error&=e_i2c_write(device_add);     // Device address
error&=e_i2c_write(reg);      // Register address

error&=e_i2c_restart();
error&=e_i2c_write(device_add+1);   // To change data direction ([bit 0]=1)
  error&=e_i2c_read(&value);     // read single byte
e_i2c_nack(); // only 1 byte is being read, so send nack
e_i2c_stop();              // end read cycle
if(error)
break;
e_i2c_reset();
}

    return value;
}

// ATTENTION: This function doesn't work
/*
char e_i2cp_read_string(char device_add, unsigned char read_buffer[], char start_address, char string_length)
{
char error=0;
char i = 0;
while(!error)
{
error=1;
error&=e_i2c_start();
error&=e_i2c_write(device_add);     // Device address
error&=e_i2c_write(start_address); // address of first register to be read
error&=e_i2c_restart();
error&=e_i2c_write(device_add+1);     // To change data direction ([bit 0]=1)

for (i=0;i < string_length;i++)
{
```

```
//   error&=e_i2c_read(&read_buffer[i]);  // read the next byte
if (i == (string_length-1)) // the last byte to be read, must send nack
error&=e_i2c_nack();
else
error&=e_i2c_ack(); // not the last byte, send ack
}
e_i2c_stop();              // End read cycle
if(error)
break;
e_i2c_reset();
}
return error;
}
*/

/*! \brief Write a specific register on a device
 * \param device_add The address of the device you want information
 * \param reg The register address you want read on the device
 * \param value The data you want to write
 * \return 1 to confirme the oparation and 0 for an error
 */
char e_i2cp_write (char device_add, char reg, char value)
{
char error=0;

while(!error)
{
error=1;
error&=e_i2c_start();
error&=e_i2c_write(device_add); // Writing the device (slave) address
error&=e_i2c_write(reg); // Device register address
error&=e_i2c_write(value); // Data to device
error&=e_i2c_stop(); // Ending the communication
if(error)
break;
e_i2c_reset();
}
return error;
}

//BOF: ne marche surement pas
/*
char e_i2cp_write_string (char device_add, unsigned char write_buffer[], char start_address, char string_length)
{
char error=0;
int i = 0;

while(!error)
{
error=1;
error&=e_i2c_start();
error&=e_i2c_write(device_add);    // Writing the device (slave) I2C address
error&=e_i2c_write(start_address);      // Device register address
for (i=0;i<string_length;i++)
error&=e_i2c_write(write_buffer[i]);       // Data to device
error&=e_i2c_stop();             // Ending the communication
if(error)
break;
// e_i2c_reset();
}
return error;
}
*/
```

2.11 matlab

2.11.1 matlab.{h,c}

```
/***************************************************************************

To communicate with matlab
August 2007: first version
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \ingroup matlab
 * \brief To communicate with matlab.
 *
 * This module manage the communication with matlab through bluetooth.
 * \author Code: Michael Bonani, Jonathan Besuchet, Doc: Jonathan Besuchet
 */

/*! \defgroup matlab Matlab communication
 *
 * \section intro_sec Introduction
 * This package contains all the ressources you need to communicate with matlab
 * through bluetooth.
 * \n \n 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.
 *
 * \subsection intro_subsec 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: \ref e_receive_int_from_matlab(int *data, int array_size).
 *
 * \subsection intro_subsec2 Sending data from e-puck
 * Now if you want to send data from e-puck to matlab you have to call the function
 * \ref e_send_int_to_matlab(int* data, int array_size) (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.
 * \sa matlab
 * \author Doc: Jonathan Besuchet
 */

#ifndef MATLAB_H
#define MATLAB_H

void e_send_int_to_matlab(int* data, int array_size);
void e_send_char_to_matlab(char * data, int array_size);
int e_receive_int_from_matlab(int* data, int array_size);
int e_receive_char_from_matlab(char* data, int array_size);

#endif // MATLAB_H
```

```
/*****************************************************************************

To communicate with matlab
August 2007: first version
Michael Bonani, Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2007 Michael Bonani, Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup matlab
 * \brief To communicate with matlab.
 *
 * This module manage the communication with matlab through bluetooth.
 * \author Code: Michael Bonani, Jonathan Besuchet, Doc: Jonathan Besuchet
 */

#include "matlab.h"

#include "./../motor_led/e_epuck_ports.h"
#include "./../uart/e_uart_char.h"

/*! \brief The function to send int values to matlab
 * \param data The array of int data you want to send
 * \param array_size The length of the array
 */
void e_send_int_to_matlab(int* data, int array_size) {
int size = 2*array_size;

e_send_uart1_char("I",1);
e_send_uart1_char((char *) &size, sizeof(int));
e_send_uart1_char((char*)data, size);
e_send_uart1_char("EOF\0",4);

return;
}

/*! \brief The function to send char values to matlab
 * \param data The array of char data you want to send
 * \param array_size The length of the array
 */
void e_send_char_to_matlab(char* data, int array_size) {
e_send_uart1_char("C",1);
e_send_uart1_char((char *) &array_size, sizeof(int));
e_send_uart1_char(data, array_size);
e_send_uart1_char("EOF\0",4);

return;
}

/*! \brief The function to receive int values from matlab
 * \param data The array of int data you want to fill
 * \param array_size The length of the array
 * \return The number of int stored
 */
int e_receive_int_from_matlab(int* data, int array_size) {
char c;
int i=0;
int temp_size=0;
int flush = 0;
```

```
do{
     if (e_getchar_uart1(&c)) {
    // The first byte is the length of the datas
     if(i==0) {
     // LSB
     temp_size = (int)c;
    }
else if(i==1) {
    // MSB
    temp_size += ((int)c)<<8;
    if((temp_size>>1) > array_size) {
    // ! We will receive more data than the buffer can store
    // Store as max data as possible, but excedded data will be flushed !
    flush = 1;
LED4 = LED4^1;
    temp_size = (array_size<<1);
}
    }
else {
    // Then the datas come
    if((i%2) == 0){
// LSB
     data[i/2-1] = (int)c;
}
else{
     // MSB
     data[(i-1)/2-1] = (data[(i-1)/2-1] & 0b0000000011111111) | ((int)c)<<8;
}
}
      i++;
}
    }
    while(i<2 || i<=(temp_size+1));

    // Flush pending data
    if(flush) {
while(e_getchar_uart1(&c));
}

    // return the number of int. stored
    return (temp_size >> 1);
}

/*! \brief The function to receive char values from matlab
 * \param data The array of char data you want to fill
 * \param array_size The length of the array
 * \return The number of char stored
 */
int e_receive_char_from_matlab(char* data, int array_size) {
char c;
int i=0;
int temp_size=0;
int flush = 0;

do{
     if (e_getchar_uart1(&c)) {
    // The first byte is the length of the datas
     if(i==0) {
     // LSB
     temp_size = (int)c;
    }
  else if(i==1) {
    // MSB
    temp_size += ((int)c)<<8;
    if(temp_size > array_size){
    // ! We will receive more data than the buffer can store
    // Store as max data as possible, but excedded data will be flushed !
    flush = 1;
    temp_size = array_size;
}
    }
else {
    // Then the datas come
    data[i-2] = c;
}
      i++;
}
    }
    while(i<2 || i<=(temp_size+1));

    // Flush pending data
    if(flush) {
while(e_getchar_uart1(&c));
}

    // return the number of char. stored
    return temp_size;
}
```

2.11.2 matlab_files/CloseEpuck.m

```
%! \brief Close the communication with the e-puck
function CloseEpuck()
global EpuckPort;
fclose(EpuckPort);
clear EpuckPort;
clear global EpuckPort;
end
```

2.11.3 matlab_files/EpuckFlush.m

```
function [] = 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, Matlab probably send old data if you don't do this
%   flush, so the e-puck could crash...

global EpuckPort;

flushinput(EpuckPort);
flushoutput(EpuckPort);
```

2.11.4 matlab_files/EpuckGetData.m

```
function [data] = EpuckGetData()
global EpuckPort;
i = 0;

while (i ~= 5)
    c = fread(EpuckPort,1,'char');
    switch i
        case 0
            if(c ~= 'E')
                continue;
            else
                i = 1;
            end
        case 1
            if(c ~= 'O')
                i = 0;
                continue;
            else
                i = 2;
            end
        case 2
            if(c ~= 'F')
                i = 0;
                continue;
            else
                i = 3;
            end
        case 3
            if(c ~= 0)
                i = 0;
                continue;
            else
                i = 4;
            end
        case 4
            if(c == 'I')
                i = 5;
                receivedFormat = c;
            elseif(c == 'C')
                i = 5;
                receivedFormat = c;
            else
                i = 0;
                continue;
            end
    end
end

if(receivedFormat == 'I')
    size = fread(EpuckPort,1,'uint16');
    data = fread(EpuckPort,size/2,'int16');
elseif(receivedFormat == 'C')
    size = fread(EpuckPort,1,'uint16');
    data = fread(EpuckPort,size,'int8');
end
return;
```

2.11.5 matlab_files/EpuckSendData.m

```
function [data] = EpuckSendData(data, dataType)
global EpuckPort;

size = length(data);

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');
end
return;
```

2.11.6 matlab_files/OpenEpuck.m

```
%! \brief Open the communication with the e-puck
 % \params port The port in wich the e-puck is paired to.
 % it must be a string like that "COM11" if e-puck is paired
 % on COM 11.
 %/
function OpenEpuck(port)
```

```
global EpuckPort;
EpuckPort = serial(port,'BaudRate', 115200,'inputBuffersize',4096,'OutputBufferSize',4096,'ByteOrder','littleendian');
try
    fopen(EpuckPort);
catch
    % could not open the port, delete the global variable
    clear EpuckPort
    clear global EpuckPort;
    disp 'Error, Could not open serial port'
    return;
end
return
```

2.12 motor_led

2.12.1 e_epuck_ports.h

```
/************************************************************************
*  Defintition of all port of the e-puck                 *
* Version 1.0 november 2005                    *
* Michael Bonani, Francesco Mondada, Davis Dadie                *
*                                  *
************************************************************************/
/*****************************************************************************

Defintition of all port of the e-puck
Version 1.0 november 2005
Michael Bonani, Francesco Mondada, Davis Dadie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Davis Dadie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Define all the usefull names corresponding of e-puck's hardware.
 * \author Code: Michael Bonani, Francesco Mondada, Davis Dadie \n Doc: Jonathan Besuchet
 */

/*! \mainpage e-puck standard library documentation
 * \image html logo.gif
 * \section intro_sec 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.
 * \n \n 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)
 * \section brief_sec 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.
 * \section link_sec 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
 */

/*! \defgroup motor_LED Ports, motors and LEDs
 *
 * \section intro_sec_motors 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.
 *
 * \subsection intro_subsecl_ports 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.
 * \n This work is made in the file: e_epuck_ports.h.
 *
 * \subsection intro_subsec2_motors 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
 *   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
 *   for more information about fast_agenda). This solution is exploited by the file
 *   library\motor_led\advance_one_timer\fast_agenda\e_motors.c.
 *
 * \subsection intro_subsec3_led 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 or e_agenda_fast.h for more
 * information about agenda or fast agenda).
 *
 * \subsection intro_subsec4_ir 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 for more information about agenda).
 *
 * \section timer_sect_timer 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
 *
 * \section agenda_sect_agenda Agenda solution
 * As we have seen, we can use the agenda solution to make the motors work.
 * \n \n So what is an agenda ?
 * \n 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.
 * \n \n How it works ?
 * \n 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.
 * \sa e_agenda.c, e_agenda_fast.c
 *
 * \author Doc: Jonathan Besuchet
 */

#ifndef _EPUCK_PORTS
#define _EPUCK_PORTS

#include "p30f6014A.h"

/*********************GENERAL SETUP************************/

#define FOSC   7.3728e6     // 7.3728Mhz crystal in XTL mode
#define PLL    8.0        // 8x PLL


#define FCY     ((FOSC*PLL)/(4.0)) // Instruction cycle frequency
#define MILLISEC  (FCY/1.0e3) // 1mSec delay constant
#define MICROSEC  (FCY/1.0e6) // 1uSec delay constant
#define NANOSEC   (FCY/1.0e9) // 1nSec delay constant

#define TCY_PIC (1e9/FCY) //time instruction cycle in [ns]
#define INTERRUPT_DELAY (10*TCY_PIC)//delay to start an interrupt in [ns] (observe with p30f6014)

#define TRUE 1
#define FALSE 0


/********************** OUTPUTS ***************************/
#define OUTPUT_PIN 0
/*LEDS*/
/*First in front of robot than turning clokwise*/
#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
```

```
/*IR*/
#define PULSE_IR0 _LATF7 // pulse IR 0 and 4
#define PULSE_IR1 _LATF8 // pulse IR 1 and 5
#define PULSE_IR2 _LATG0 // pulse IR 2 and 6
#define PULSE_IR3 _LATG1 // pulse IR 3 and 7

#define PULSE_IR0_DIR _TRISF7
#define PULSE_IR1_DIR _TRISF8
#define PULSE_IR2_DIR _TRISG0
#define PULSE_IR3_DIR _TRISG1

/*First in front right of robot than turning clokwise*/
#define IR0 8  // ir proximity sensor 0 on AD channel 8
#define IR1 9  // ir proximity sensor 1 on AD channel 9
#define IR2 10  // ir proximity sensor 2 on AD channel 10
#define IR3 11  // ir proximity sensor 3 on AD channel 11
#define IR4 12  // ir proximity sensor 4 on AD channel 12
#define IR5 13  // ir proximity sensor 5 on AD channel 13
#define IR6 14  // ir proximity sensor 6 on AD channel 14
#define IR7 15  // ir proximity sensor 7 on AD channel 15


/*analog*/
#define MIC1 2  // microphone 1 on AD channel 2
#define MIC2 3  // microphone 2 on AD channel 3
#define MIC3 4  // microphone 3 on AD channel 4


#define ACCX 5  // X Axis of accelerometer on AD channel 5
#define ACCY 6  // Y Axis of accelerometer on AD channel 6
#define ACCZ 7  // Z Axis of accelerometer on AD channel 7


/*basic audio*/
#define AUDIO_ON _LATF0
#define AUDIO_ON_DIR _TRISF0

/*motors*/
#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

/*camera*/
#define CAM_RESET _LATC13
#define CAM_RESET_DIR _TRISC13

/* I2C */
#define SIO_D _LATG3
#define SIO_D_DIR _TRISG3

#define SIO_C _LATG2
#define SIO_C_DIR _TRISG2

/********************** INPUTS ************************/
#define INPUT_PIN 1

/*low battery signal active low when Vbatt<3.4V*/
#define BATT_LOW _RF1
#define BATT_LOW_DIR _TRISF1

/* selector on normal extension*/
#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

/*IR TV receiver on normal extension*/
#define REMOTE _RF6
#define REMOTE_DIR _TRISF6

/*CAMERA*/
/*data higher 8 bits of port D*/
#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

/*clock interupt*/
#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

/********************** ASEMBLER SMALL FUNCTCION********************** */
#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

#endif
```

2.12.2 e_init_port.{h,c}

```
/*****************************************************************************

Initialization of port of e-puck
Version 1.0 november 2005
Michael Bonani, Francesco Mondada, Davis Dadie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Davis Dadie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Initialize the ports on standard configuration.
 * \author Code: Michael Bonani, Francesco Mondada, Davis Dadie \n Doc: Jonathan Besuchet
 */

#ifndef _INIT_PORT
#define _INIT_PORT

/* functions */
void e_init_port(void);

#endif
```

```
/*****************************************************************************

Initialization of port of e-puck
Version 1.0 november 2005
Michael Bonani, Francesco Mondada, Davis Dadie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Davis Dadie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Initialize the ports on standard configuration.
 * \author Code: Michael Bonani, Francesco Mondada, Davis Dadie \n Doc: Jonathan Besuchet
 */

#include "e_epuck_ports.h"

/* set configuration bit for MPLAB
!!!remove this macros if you have want different configuration bit!!!*/
_FOSC(CSW_FSCM_OFF & XT_PLL8);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & MCLR_EN);
```

```
_FGS(CODE_PROT_OFF);

/*! \brief 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).
 */
void e_init_port(void)
{

/********************* OUTPUTS **************************/
/*LEDS*/
LED0 = 0;
LED1 = 0;
LED2 = 0;
LED3 = 0;
LED4 = 0;
LED5 = 0;
LED6 = 0;
LED7 = 0;
LED0_DIR = OUTPUT_PIN;
LED1_DIR = OUTPUT_PIN;
LED2_DIR = OUTPUT_PIN;
LED3_DIR = OUTPUT_PIN;
LED4_DIR = OUTPUT_PIN;
LED5_DIR = OUTPUT_PIN;
LED6_DIR = OUTPUT_PIN;
LED7_DIR = OUTPUT_PIN;

FRONT_LED = 0;
FRONT_LED_DIR = OUTPUT_PIN;

BODY_LED = 0;
BODY_LED_DIR = OUTPUT_PIN;

/*IR*/
PULSE_IR0 = 0;
PULSE_IR1 = 0;
PULSE_IR2 = 0;
PULSE_IR3 = 0;
PULSE_IR0_DIR = OUTPUT_PIN;
PULSE_IR1_DIR = OUTPUT_PIN;
PULSE_IR2_DIR = OUTPUT_PIN;
PULSE_IR3_DIR = OUTPUT_PIN;

/*basic audio*/
AUDIO_ON = 0; /*turn of speaker and codec*/
AUDIO_ON_DIR = OUTPUT_PIN;

/*motors*/
MOTOR1_PHA = 0;
MOTOR1_PHB = 0;
MOTOR1_PHC = 0;
MOTOR1_PHD = 0;
MOTOR2_PHA = 0;
MOTOR2_PHB = 0;
MOTOR2_PHC = 0;
MOTOR2_PHD = 0;
MOTOR1_PHA_DIR = OUTPUT_PIN;
MOTOR1_PHB_DIR = OUTPUT_PIN;
MOTOR1_PHC_DIR = OUTPUT_PIN;
MOTOR1_PHD_DIR = OUTPUT_PIN;
MOTOR2_PHA_DIR = OUTPUT_PIN;
MOTOR2_PHB_DIR = OUTPUT_PIN;
MOTOR2_PHC_DIR = OUTPUT_PIN;
MOTOR2_PHD_DIR = OUTPUT_PIN;

/*camera*/
CAM_RESET=0;
CAM_RESET_DIR = OUTPUT_PIN;

/*I2C*/
SIO_C=0;
SIO_D=0;

SIO_C_DIR= OUTPUT_PIN;
SIO_D_DIR= OUTPUT_PIN;

/********************* INPUTS **************************/

/*low battery signal active low when Vbatt<3.4V*/
BATT_LOW_DIR = INPUT_PIN;

/*IR TV receiver on normal extension*/
REMOTE_DIR = INPUT_PIN;

/* selector*/
SELECTOR0_DIR = INPUT_PIN;
SELECTOR1_DIR = INPUT_PIN;
SELECTOR2_DIR = INPUT_PIN;
SELECTOR3_DIR = INPUT_PIN;

/*camera*/
CAM_y0_DIR = INPUT_PIN;
CAM_y1_DIR = INPUT_PIN;
CAM_y2_DIR = INPUT_PIN;
CAM_y3_DIR = INPUT_PIN;
CAM_y4_DIR = INPUT_PIN;
CAM_y5_DIR = INPUT_PIN;
CAM_y6_DIR = INPUT_PIN;
CAM_y7_DIR = INPUT_PIN;
}
```

2.12.3 e_led.{h,c}

```
/***************************************************************************

fonctions for simple LED manipulation
Version 1.0 november 2005
Michael Bonani, Francesco Mondada, Davis Dadie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Davis Dadie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the LEDs.
 * \n A little exemple for the LEDs (all the LEDs are blinking)
 * \code
 * #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
 *  }
 * }
 * \endcode
 * \author Code: Michael Bonani, Francesco Mondada, Davis Dadie \n Doc: Jonathan Besuchet
 */

#ifndef _LED
#define _LED

/* functions */
void e_set_led(unsigned int led_number, unsigned int value); // set led_number (0-7) to value (0=off 1=on higher=inverse)
void e_led_clear(void);

void e_set_body_led(unsigned int value); // value (0=off 1=on higher=inverse)
void e_set_front_led(unsigned int value); //value (0=off 1=on higher=inverse)

#endif
```

```
/***************************************************************************

fonctions for simple LED manipulation
Version 1.0 november 2005
Michael Bonani, Francesco Mondada, Davis Dadie


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Davis Dadie

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the LEDs.
 * \n A little exemple for the LEDs (all the LEDs are blinking)
 * \code
 * #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
 *  }
 * }
 * \endcode
 * \author Code: Michael Bonani, Francesco Mondada, Davis Dadie \n Doc: Jonathan Besuchet
 */
```

```
#include "e_epuck_ports.h"
#include "e_led.h"

/*! \brief turn on/off the specified LED
 *
 * The e-puck has 8 green LEDs. With this function, you can
 * change the state of these LEDs.
 * \param led_number between 0 and 7
 * \param 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.
 */
void e_set_led(unsigned int led_number, unsigned int value)
{
switch(led_number)
{
case 0:
{
if(value>1)
LED0 = LED0^1; // "^" exclusif OR bit to bit
else
LED0 = value;
break;
}
case 1:
{
if(value>1)
LED1 = LED1^1;
else
LED1 = value;
break;
}
case 2:
{
if(value>1)
LED2 = LED2^1;
else
LED2 = value;
break;
}
case 3:
{
if(value>1)
LED3 = LED3^1;
else
LED3 = value;
break;
}
case 4:
{
if(value>1)
LED4 = LED4^1;
else
LED4 = value;
break;
}
case 5:
{
if(value>1)
LED5 = LED5^1;
else
LED5 = value;
break;
}
case 6:
{
if(value>1)
LED6 = LED6^1;
else
LED6 = value;
break;
}
case 7:
{
if(value>1)
LED7 = LED7^1;
else
LED7 = value;
break;
}
default:
if(value > 1)
{
LED0 = LED0^1; LED1 = LED1^1; LED2 = LED2^1; LED3 = LED3^1;
LED4 = LED4^1; LED5 = LED5^1; LED6 = LED6^1; LED7 = LED7^1;
}
else
LED0 = LED1 = LED2 = LED3 = LED4 = LED5 = LED6 = LED7 = value;
}
}

/*! \brief 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.
 * \param value 0 (off), 1 (on) otherwise change the state
 */
void e_set_body_led(unsigned int value)
{
if(value>1)
BODY_LED = BODY_LED^1;
else
BODY_LED = value;
}

/*! \brief 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.
 * \param value 0 (off), 1 (on) otherwise change the state
 */
void e_set_front_led(unsigned int value)
{
if(value>1)
FRONT_LED = FRONT_LED^1;
else
FRONT_LED = value;
}

/*! \brief 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".
 */
void e_led_clear(void)
{
LED0 = 0;
LED1 = 0;
LED2 = 0;
LED3 = 0;
LED4 = 0;
LED5 = 0;
LED6 = 0;
LED7 = 0;
}
```

2.12.4 e_motors_timer3.c

```
/*****************************************************************************

Control motor of e-puck with timer 3
December 2004: first version
Lucas Meier & Francesco Mondada
Version 1.0 november 2005
Michael Bonani
Version 1.1 january 2006
Xavier Raemy


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Lucas Meier, Xavier Raemy

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Initialize the ports on standard configuration.
 * \author Code: Michael Bonani, Francesco Mondada, Davis Dadie \n Doc: Jonathan Besuchet
 */
/*! \file
 * \ingroup motor_LED
 * \brief Manage the motors (with timer3).
 *
 * This module manage the two motors with one timer: timer3.
 * \author Code: Michael Bonani, Francesco Mondada, Lucas Meier, Xavier Raemy \n Doc: Jonathan Besuchet
 */

#include <stdlib.h>
#include "e_epuck_ports.h"
#include "e_init_port.h"

/* internal 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;

/* internal calls */

void __attribute__((interrupt, auto_psv, shadow))
 _T3Interrupt(void) // interrupt for motor
{
  static int motor_phase_left=0;  // phase can be 0 to 3
  static int motor_phase_right=0;

  IFS0bits.T3IF = 0;             // clear interrupt flag


  if (left_speed != 0)
   motor_counter_left--;
  else
  {
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
    MOTOR1_PHC = 0;
    MOTOR1_PHD = 0;
  }
  if (right_speed != 0)
   motor_counter_right--;
```

```
  else
  {
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
    MOTOR2_PHC = 0;
    MOTOR2_PHD = 0;
  }

  if (motor_counter_left <= 0)
  {
motor_counter_left=motor_counter_left_init;

   // increment or decrement phase depending on direction

   if (left_speed > 0) // inverted for the two motors
    {
     nbr_pas_left++;
motor_phase_left--;
     if (motor_phase_left < 0) motor_phase_left = 3;
    }
   else
    {
     nbr_pas_left--;
     motor_phase_left++;
    if (motor_phase_left > 3) motor_phase_left = 0;
    }

   // set the phase on the port pins

  switch (motor_phase_left)
   {
    case 0:
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    case 1:
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    case 2:
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    case 3:
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
   }
  }


  if (motor_counter_right <= 0)
  {
motor_counter_right=motor_counter_right_init;
   if (right_speed < 0)
    {
      nbr_pas_right--;
      motor_phase_right--;
      if (motor_phase_right < 0) motor_phase_right = 3;
    }
    else
    {
      nbr_pas_right++;
      motor_phase_right++;
      if (motor_phase_right > 3) motor_phase_right = 0;
    }

   // set the phase on the port pins

   switch (motor_phase_right)
   {
    case 0:
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    case 1:
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    case 2:
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    case 3:
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
   }
  }
```

```
}


/* ---- user calls ---- */

/*! \brief Give the number of left motor steps
 * \return The number of phases steps made since the left motor
 * is running.
 */
int e_get_steps_left(void)
{
  return nbr_pas_left;
}

/*! \brief Set the number of left motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_left(int set_steps)
{
  INTERRUPT_OFF();
  nbr_pas_left = set_steps;
  INTERRUPT_ON();
}

/*! \brief Give the number of right motor steps
 * \return The number of phases steps made since the right motor
 * is running.
 */
int e_get_steps_right(void)
{
  return nbr_pas_right;
}

/*! \brief Set the number of right motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_right(int set_steps)
{
  INTERRUPT_OFF();
  nbr_pas_right = set_steps;
  INTERRUPT_ON();
}

/*! \brief Manage the left speed
 *
 * This function manage the left motor speed by changing the MOTOR1
 * phases. The changing phases frequency (=> speed) is controled by
 * the timer3.
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 */
void e_set_speed_left(int motor_speed)  // motor speed in steps/s
{
INTERRUPT_OFF();
if (motor_speed != 0)
{
if (motor_speed > 999)
motor_speed = 999;
if (motor_speed <= -999)
motor_speed = -999;
motor_counter_left=abs(10000/motor_speed);
}
else
motor_counter_left=9999;
motor_counter_left_init = motor_counter_left;
left_speed = motor_speed;
INTERRUPT_ON();
}

/*! \brief Manage the right speed
 *
 * This function manage the right motor speed by changing the MOTOR2
 * phases. The changing phases frequency (=> speed) is controled by
 * the timer3.
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 */
void e_set_speed_right(int motor_speed)  // motor speed in steps/s
{
INTERRUPT_OFF();
if (motor_speed != 0)
{
if (motor_speed > 999)
motor_speed = 999;
if (motor_speed < -999)
motor_speed = -999;
motor_counter_right=abs(10000/motor_speed);
}
else
motor_counter_right=9999;
motor_counter_right_init = motor_counter_right;
right_speed = motor_speed;
INTERRUPT_ON();
}

/*! \brief Initialize the motors's ports
 *
 * This function configure timer3 and initialize the ports
 * used by the motors. In fact it call "the e_init_port()" function.
 * \sa e_init_port
 */
void e_init_motors(void)
{
    T3CONbits.TON = 0;            // stop Timer3
   e_init_port();   // init general ports
    left_speed = 0;
right_speed = 0;
```

```
motor_counter_left_init=9999;
motor_counter_right_init=9999;
motor_counter_left=9999;
motor_counter_right=9999;

    T3CON = 0;                      //
    T3CONbits.TCKPS=3;              // prescsaler = 256
    TMR3 = 0;                       // clear timer 3
    PR3 = (FCY/256)/10000;     // interrupt every 0.1ms with 256 prescaler
    IFS0bits.T3IF = 0;              // clear interrupt flag
    IEC0bits.T3IE = 1;              // set interrupt enable bit
    T3CONbits.TON = 1;              // start Timer3
}
```

2.12.5 e_motors.{h,c}

```
/*****************************************************************************

Control motor of e-puck with timer 4 and 5
December 2004: first version
Lucas Meier & Francesco Mondada
Version 1.0 november 2005
Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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)
 * \code
 * #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) {}
 * }
 * \endcode
 * \author Code: Michael Bonani, Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include <stdlib.h>
#include "e_epuck_ports.h"
#include "e_init_port.h"
#include "e_motors.h"

/* internal variables */

static int left_speed = 0;
static int right_speed = 0;
static int nbr_pas_left = 0;
static int nbr_pas_right = 0;

/* internal calls */

void __attribute__((interrupt, auto_psv, shadow))
 _T5Interrupt(void) // interrupt for motor 1 (of two) = left motor
{
  static int motor_phase=0;  // phase can be 0 to 3

  IFS1bits.T5IF = 0;             // clear interrupt flag

  // increment or decrement phase depending on direction

  if (left_speed > 0) // inverted for the two motors
    {
    nbr_pas_left++;
motor_phase--;
    if (motor_phase < 0) motor_phase = 3;
    }
  else
    {
    nbr_pas_left--;
    motor_phase++;
    if (motor_phase > 3) motor_phase = 0;
    }

  // set the phase on the port pins

  switch (motor_phase)
```

```
  {
    case 0:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
  }
}

void __attribute__((interrupt, auto_psv, shadow))
 _T4Interrupt(void)  // interrupt for motor 2 (of two) = right motor
{
  static int motor_phase=0; // phase can be 0 to 3

  IFS1bits.T4IF = 0;             // clear interrupt flag

  // increment or decrement phase depending on direction

  if (right_speed < 0)
    {
    nbr_pas_right--;
    motor_phase--;
    if (motor_phase < 0) motor_phase = 3;
    }
  else
    {
    nbr_pas_right++;
    motor_phase++;
    if (motor_phase > 3) motor_phase = 0;
    }

  // set the phase on the port pins

  switch (motor_phase)
  {
    case 0:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
  }
}


/* ---- user calls ---- */

/*! \brief Initialize the motors's ports
 *
 * This function initialize the ports used by the motors.
 * In fact it call "the e_init_port()" function.
 * \sa e_init_port
 */
void e_init_motors(void)
```

```
{
  e_init_port(); // init general ports
}

/*! \brief Manage the left speed
 *
 * This function manage the left motor speed by changing the MOTOR1
 * phases. The changing phases frequency (=> speed) is controled by
 * the timer5.
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 */
void e_set_speed_left(int motor_speed)  // motor speed in steps/s
{
  if (motor_speed == 0)
  {
    T5CONbits.TON = 0;            // stop Timer5
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
    MOTOR1_PHC = 0;
    MOTOR1_PHD = 0;
  }
  else
  {
    T5CONbits.TON = 0;            // stop Timer5

    left_speed = motor_speed;

    T5CON = 0;                    //
    T5CONbits.TCKPS=3;            // prescsaler = 256
    TMR5 = 0;                     // clear timer 5
    PR5 = (FCY/256)/abs(motor_speed);  // interrupt every Xms with 256 prescaler
    IFS1bits.T5IF = 0;            // clear interrupt flag
    IEC1bits.T5IE = 1;            // set interrupt enable bit
    T5CONbits.TON = 1;            // start Timer5
  }
}

/*! \brief Manage the right speed
 *
 * This function manage the right motor speed by changing the MOTOR2
 * phases. The changing phases frequency (=> speed) is controled by
 * the timer4.
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 */
void e_set_speed_right(int motor_speed)
{
  if (motor_speed == 0)
  {
    T4CONbits.TON = 0;            // stop Timer4
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
    MOTOR2_PHC = 0;
    MOTOR2_PHD = 0;
  }
  else
  {
    T4CONbits.TON = 0;            // stop Timer4

    right_speed = motor_speed;

    T4CON = 0;                    //
    T4CONbits.TCKPS=3;            // prescsaler = 256
    TMR4 = 0;                     // clear timer 4
    PR4 = (FCY/256)/abs(motor_speed);  // interrupt every Xms with 256 prescaler
    IFS1bits.T4IF = 0;            // clear interrupt flag
    IEC1bits.T4IE = 1;            // set interrupt enable bit
    T4CONbits.TON = 1;            // start Timer4
  }
}

/*! \brief Give the number of left motor steps
 * \return The number of phases steps made since the left motor
 * is running.
 */
int e_get_steps_left(void)
{
  return nbr_pas_left;
}

/*! \brief Set the number of left motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_left(int set_steps)
{
  INTERRUPT_OFF();
  nbr_pas_left = set_steps;
  INTERRUPT_ON();
}

/*! \brief Give the number of right motor steps
 * \return The number of phases steps made since the right motor
 * is running.
 */
int e_get_steps_right(void)
{
  return nbr_pas_right;
}

/*! \brief Set the number of right motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_right(int set_steps)
{
  INTERRUPT_OFF();
  nbr_pas_right = set_steps;
```

```
  INTERRUPT_ON();
}
```

```
/*************************************************************************

Control motor of e-puck with timer 4 and 5
December 2004: first version
Lucas Meier & Francesco Mondada
Version 1.0 november 2005
Michael Bonani


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2005-2007 Michael Bonani, Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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)
 * \code
 * #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) {}
 * }
 * \endcode
 * \author Code: Michael Bonani, Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef _MOTORS
#define _MOTORS

/* functions */

void e_init_motors(void); // init to be done before using the other calls

void e_set_speed_left(int motor_speed);  // motor speed in steps/s
void e_set_speed_right(int motor_speed);   // motor speed in steps/s
int e_get_steps_left(void); // motors steps done left
int e_get_steps_right(void); // motors steps done right
void e_set_steps_left(int set_steps); // set motor steps counter
void e_set_steps_right(int set_steps); // set motor steps counter
#endif
```

2.13 motor_led/advance_one_timer

2.13.1 e_agenda_NU.{h,c}

```
/*************************************************************************

Advance agenda events of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the agendas (timer2)
 *
 * This module manage the agendas with the timer2.
 * \n \n 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.
 * \n \n 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.
 * \n \n If one (or more) agenda has to be treated, his callback function is launch.
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef __AGENDA_H__
#define __AGENDA_H__


#define AG_ALREADY_CREATED 1
#define AG_NOT_FOUND 2

/*****************************************************************
 * ----------------------- Type definition --------------------------
 *****************************************************************/

typedef struct AgendaType Agenda;

/*! \struct AgendaType
 * \brief 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".
 * \n The member "activate" can be on=1 or off=0. When it is off, the counter
 * don't increase.
 * \n This struct is designed to be used as chained list so we need a pointer
 * to the next element.
 */
struct AgendaType
{
  unsigned int  cycle; /*!< length in 10e of ms of a cycle between two events */
  int  counter; /*!< count the number of interrupts */
  char activate; /*!< can be on=1 or off=0*/
  void (*function) (void); /*!< function called when counter > cycle,
                            * \warning This function must have the following
                            * prototype: "void func(void)" */
  Agenda *next; /*!< pointer on the next agenda*/
};


/*****************************************************************
 * ----------------------- From agenda.c file -------------------------
 *****************************************************************/
void e_start_agendas_processing(void);
void e_end_agendas_processing(void);

int e_activate_agenda(void (*func)(void), int cycle);
int e_destroy_agenda(void (*func)(void));

int e_set_agenda_cycle(void (*func)(void), int cycle);
int e_reset_agenda(void (*func)(void));

int e_pause_agenda(void (*func)(void));
int e_restart_agenda(void (*func)(void));

#endif /* __AGENDA_H__ */


/* End of File : agenda.h */
```

```
/*****************************************************************************

Advance agenda events of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada

Modified Dec. 2007 by Kevin Lynch to increase the time between interrupts.
That's the only modification from e_agenda.c.


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the agendas (timer2)
 *
 * This module manage the agendas with the timer2.
 * \n \n 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.
 * \n \n 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.
 * \n \n If one (or more) agenda has to be treated, his callback function is launch.
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include "e_agenda_NU.h"
#include "../e_epuck_ports.h"
#include <stdlib.h>

#define EXIT_OK 1
```

```
/*!pointer on the end of agenda chained list */
static Agenda *agenda_list = 0;


/*! \brief Start the agendas processing
 *
 * Start the agendas processing by starting the Timer2.
 */
void e_start_agendas_processing(void)
{
T2CON = 0; // reset Timer2 CONtrol register
T2CONbits.TCKPS = 0; // precsaler = 1
TMR2 = 0; // clear timer 2
PR2 = (int)(0.5*MILLISEC);     // interrupt every 0.5 ms with 64 prescaler, formerly 0.1 ms
IFS0bits.T2IF = 0; // clear interrupt flag
IEC0bits.T2IE = 1; // set interrupt enable bit
T2CONbits.TON = 1; // start Timer2
}


/*! \brief Stop all the agendas
 *
 * Stop all the agendas by disabling Timer2
 * \warning the memory allocated for the agenda isn't freed,
 * use \ref e_destroy_agenda(void (*func)(void)) for that.
 * \sa e_destroy_agenda
 */
void e_end_agendas_processing(void)
{
T2CONbits.TON = 0;    // disable Timer2
}


/*! \brief 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).
 * \param func  function called if the cycle value is reached by the counter
 * \param cycle      cycle value in millisec/10
 * \return \ref EXIT_OK if the agenda has been created, exit the programme otherwise
 */
int e_activate_agenda(void (*func)(void), int cycle)
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
return(AG_ALREADY_CREATED);
else
current = current->next;
}
if(!(current = malloc(sizeof(Agenda))))
exit (EXIT_FAILURE);

current->cycle = cycle;
current->counter = 0;
current->activate = 1;
current->function = func;
current->next = agenda_list;

agenda_list = current;
return(EXIT_OK);
}


/*! \brief Destroy an agenda
 *
 * Destroy the agenda with a given callback function.
 * \param func  function to test
 * \return \ref EXIT_OK if the agenda has been destroyed, \ref AG_NOT_FOUND otherwise
 */
int e_destroy_agenda(void (*func)(void))
{
Agenda *preceding = 0;
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
if (preceding)
preceding->next = current->next;
else
agenda_list = current->next;

free(current);
return(EXIT_OK);
}
else
{
preceding = current;
current = current->next;
}
}
return(AG_NOT_FOUND);
}


/*! \brief Change the cycle value of an agenda
 *
 * Change the cycle value of an agenda with a given callback function.
 * \param func  function to test
 * \param cycle      new cycle value in millisec/10
```

```
 * \return \ref EXIT_OK if the cycle of the agenda has been modified,
 *         \ref AG_NOT_FOUND otherwise
 */
int e_set_agenda_cycle(void (*func)(void), int cycle)
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->cycle = cycle;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);
}


/*! \brief  Reset an agenda's counter
 *
 * Reset an agenda's counter with a given callback function.
 * \param func  function to reset
 * \return \ref EXIT_OK if the cycle of the agenda has been reseted,
 *         \ref AG_NOT_FOUND otherwise
 * \warning This function RESET the agenda, if you just want a pause tell
 * \ref e_pause_agenda(void (*func)(void))
 * \sa e_pause_agenda
 */
int e_reset_agenda(void (*func)(void))
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->counter = 0;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);
}

/*! \brief Pause an agenda
 *
 * Pause an agenda but do not reset its information.
 * \param func  function to pause
 * \return \ref EXIT_OK the agenda has been paused,
 *         \ref AG_NOT_FOUND otherwise
 */
int e_pause_agenda(void (*func)(void))
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->activate = 0;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);

}

/*! \brief Restart an agenda previously paused
 *
 * Restart an agenda previously paused.
 * \param func  function to restart
 * \return \ref EXIT_OK if he agenda has been restarted,
 *         \ref AG_NOT_FOUND otherwise
 * \sa e_pause_agenda
 */
int e_restart_agenda(void (*func)(void))
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->activate = 1;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);

}


/*! \brief Interrupt from timer2
 *
 * Parse the chained list of agenda.
 * \n Increment counter only.
 * \n Check if agenda has to be treated according to the cycle value
 * and current counter value.
 * \n Do it for number of cycle positive or null.
```

```
 * \n Check if a service has to be activated.
 */
void __attribute__((interrupt, auto_psv))
 _T2Interrupt(void)
{
Agenda *current = agenda_list;

IFS0bits.T2IF = 0;

while (current)
{
// agenda must be active with a positive non-null cycle value
if(current->activate == 1 && current->cycle > 0)
{
current->counter++;
// check if the agenda event must be triggered
if(current->counter > current->cycle-1) // a cycle value of 1 will be performed every interupt
{
current->function(); // trigger the associeted function
current->counter=0; // reset the counter
}
}
current = current->next;
}
  return;
}
```

2.13.2 e_agenda.{h,c}

```
/*****************************************************************************

Advance agenda events of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the agendas (timer2)
 *
 * This module manage the agendas with the timer2.
 * \n \n 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.
 * \n \n 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.
 * \n \n If one (or more) agenda has to be treated, his callback function is launch.
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef __AGENDA_H__
#define __AGENDA_H__


#define AG_ALREADY_CREATED 1
#define AG_NOT_FOUND 2

/*****************************************************************
 * ----------------------- Type definition -------------------------
 *****************************************************************/

typedef struct AgendaType Agenda;

/*! \struct AgendaType
 * \brief 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".
 * \n The member "activate" can be on=1 or off=0. When it is off, the counter
 * don't increase.
 * \n This struct is designed to be used as chained list so we need a pointer
 * to the next element.
 */
struct AgendaType
{
  unsigned int  cycle; /*!< length in 10e of ms of a cycle between two events */
  int  counter; /*!< count the number of interrupts */
  char activate; /*!< can be on=1 or off=0*/
  void (*function) (void); /*!< function called when counter > cycle,
                            * \warning This function must have the following
                            * prototype: "void func(void)" */
  Agenda *next; /*!< pointer on the next agenda*/
};


/*****************************************************************
 * ----------------------- From agenda.c file -------------------------
 *****************************************************************/
void e_start_agendas_processing(void);
void e_end_agendas_processing(void);

int e_activate_agenda(void (*func)(void), int cycle);
```

```
int e_destroy_agenda(void (*func)(void));

int e_set_agenda_cycle(void (*func)(void), int cycle);
int e_reset_agenda(void (*func)(void));

int e_pause_agenda(void (*func)(void));
int e_restart_agenda(void (*func)(void));

#endif /* __AGENDA_H__ */


/* End of File : agenda.h */
```

```
/*****************************************************************************

Advance agenda events of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the agendas (timer2)
 *
 * This module manage the agendas with the timer2.
 * \n \n 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.
 * \n \n 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.
 * \n \n If one (or more) agenda has to be treated, his callback function is launch.
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include "e_agenda.h"
#include "../e_epuck_ports.h"
#include <stdlib.h>

#define EXIT_OK 1

/*!pointer on the end of agenda chained list */
static Agenda *agenda_list = 0;


/*! \brief Start the agendas processing
 *
 * Start the agendas processing by starting the Timer2.
 */
void e_start_agendas_processing(void)
{
T2CON = 0; // reset Timer2 CONtrol register
T2CONbits.TCKPS = 0; // precsaler = 1
TMR2 = 0; // clear timer 2
PR2 = (int)(0.1*MILLISEC); // interrupt every 0.1 ms with 64 prescaler
IFS0bits.T2IF = 0; // clear interrupt flag
IEC0bits.T2IE = 1; // set interrupt enable bit
T2CONbits.TON = 1; // start Timer2
}


/*! \brief Stop all the agendas
 *
 * Stop all the agendas by disabling Timer2
 * \warning the memory allocated for the agenda isn't freed,
 * use \ref e_destroy_agenda(void (*func)(void)) for that.
 * \sa e_destroy_agenda
 */
void e_end_agendas_processing(void)
{
T2CONbits.TON = 0;    // disable Timer2
}


/*! \brief 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).
 * \param func  function called if the cycle value is reached by the counter
 * \param cycle      cycle value in millisec/10
 * \return \ref EXIT_OK if the agenda has been created, exit the programme otherwise
 */
int e_activate_agenda(void (*func)(void), int cycle)
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
```

```
return(AG_ALREADY_CREATED);
else
current = current->next;
}
if(!(current = malloc(sizeof(Agenda))))
exit (EXIT_FAILURE);

current->cycle = cycle;
current->counter = 0;
current->activate = 1;
current->function = func;
current->next = agenda_list;

agenda_list = current;
return(EXIT_OK);
}


/*! \brief Destroy an agenda
 *
 * Destroy the agenda with a given callback function.
 * \param func  function to test
 * \return \ref EXIT_OK if the agenda has been destroyed, \ref AG_NOT_FOUND otherwise
 */
int e_destroy_agenda(void (*func)(void))
{
Agenda *preceding = 0;
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
if (preceding)
preceding->next = current->next;
else
agenda_list = current->next;

free(current);
return(EXIT_OK);
}
else
{
preceding = current;
current = current->next;
}
}
return(AG_NOT_FOUND);
}


/*! \brief Change the cycle value of an agenda
 *
 * Change the cycle value of an agenda with a given callback function.
 * \param func  function to test
 * \param cycle      new cycle value in millisec/10
 * \return \ref EXIT_OK if the cycle of the agenda has been modified,
 *         \ref AG_NOT_FOUND otherwise
 */
int e_set_agenda_cycle(void (*func)(void), int cycle)
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->cycle = cycle;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);
}


/*! \brief  Reset an agenda's counter
 *
 * Reset an agenda's counter with a given callback function.
 * \param func  function to reset
 * \return \ref EXIT_OK if the cycle of the agenda has been reseted,
 *         \ref AG_NOT_FOUND otherwise
 * \warning This function RESET the agenda, if you just want a pause tell
 * \ref e_pause_agenda(void (*func)(void))
 * \sa e_pause_agenda
 */
int e_reset_agenda(void (*func)(void))
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->counter = 0;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);
}

/*! \brief Pause an agenda
 *
 * Pause an agenda but do not reset its information.
```

```
 * \param func  function to pause
 * \return \ref EXIT_OK the agenda has been paused,
 *         \ref AG_NOT_FOUND otherwise
 */
int e_pause_agenda(void (*func)(void))
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->activate = 0;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);

}

/*! \brief Restart an agenda previously paused
 *
 * Restart an agenda previously paused.
 * \param func  function to restart
 * \return \ref EXIT_OK if he agenda has been restarted,
 *         \ref AG_NOT_FOUND otherwise
 * \sa e_pause_agenda
 */
int e_restart_agenda(void (*func)(void))
{
Agenda *current = agenda_list;

while (current)
{
if (current->function == func)
{
current->activate = 1;
return(EXIT_OK);
}
else
current = current->next;
}
return(AG_NOT_FOUND);

}


/*! \brief Interrupt from timer2
 *
 * Parse the chained list of agenda.
 * \n Increment counter only.
 * \n Check if agenda has to be treated according to the cycle value
 * and current counter value.
 * \n Do it for number of cycle positive or null.
 * \n Check if a service has to be activated.
 */
void __attribute__((interrupt, auto_psv))
 _T2Interrupt(void)
{
Agenda *current = agenda_list;

IFS0bits.T2IF = 0;

while (current)
{
// agenda must be active with a positive non-null cycle value
if(current->activate == 1 && current->cycle > 0)
{
current->counter++;
// check if the agenda event must be triggered
if(current->counter > current->cycle-1) // a cycle value of 1 will be performed every interupt
{
current->function(); // trigger the associeted function
current->counter=0; // reset the counter
}
}
current = current->next;
}
  return;
}

/* End of File : alarm.c */
```

2.13.3 e_led.{h,c}

```
/***************************************************************************

Advance control led of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada
August 2007: Led effects added
Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier
(c) 2007 Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
```

```
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

********************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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 \ref e_blink_led(void) function.
 * \code
 * #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) {}
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

#ifndef _LED
#define _LED

/* functions */
void e_set_led(unsigned int led_number, unsigned int value); // set led_number (0-7) to value (0-1)
void e_led_clear(void);
void e_blink_led(void);
void e_blink_led0(void);
void e_blink_led1(void);
void e_blink_led2(void);
void e_blink_led3(void);
void e_blink_led4(void);
void e_blink_led5(void);
void e_blink_led6(void);
void e_blink_led7(void);

void e_set_body_led(unsigned int value); // value (0=off 1=on higher=inverse)
void e_set_front_led(unsigned int value); //value (0=off 1=on higher=inverse)

void e_start_led_blinking(int cycle);
void e_stop_led_blinking(void);

void flow_led(void);
void snake_led(void);
void k2000_led(void);
void right_led(void);
void left_led(void);

#endif
```

```
/*********************************************************************************

Advance control led of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada
August 2007: Led effects added
Jonathan Besuchet


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier
(c) 2007 Jonathan Besuchet

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

********************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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 \ref e_blink_led(void) function.
 * \code
 * #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) {}
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier, Jonathan Besuchet \n Doc: Jonathan Besuchet
 */

#include "../e_epuck_ports.h"
#include "e_agenda.h"

/*! \brief For the preprocessor.
 * Comment if you want not to use the LED effects.
 */
#define LED_EFFECTS

/*! \brief turn on/off the specified LED
 *
 * The e-puck has 8 green LEDs. With this function, you can
 * change the state of these LEDs.
 * \param led_number between 0 and 7
 * \param 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.
 */
void e_set_led(unsigned int led_number, unsigned int value)
{
switch(led_number)
{
case 0:
{
if(value>1)
LED0 = LED0^1;
else
LED0 = value;
break;
}
case 1:
{
if(value>1)
LED1 = LED1^1;
else
LED1 = value;
break;
}
case 2:
{
if(value>1)
LED2 = LED2^1;
else
LED2 = value;
break;
}
case 3:
{
if(value>1)
LED3 = LED3^1;
else
LED3 = value;
break;
}
case 4:
{
if(value>1)
LED4 = LED4^1;
else
LED4 = value;
break;
}
case 5:
{
if(value>1)
LED5 = LED5^1;
else
LED5 = value;
break;
}
case 6:
{
if(value>1)
LED6 = LED6^1;
else
LED6 = value;
break;
}
case 7:
{
if(value>1)
LED7 = LED7^1;
else
LED7 = value;
break;
}
default:
LED0 = LED1 = LED2 = LED3 = LED4 = LED5 = LED6 = LED7 = value;
}
}

/*! \brief 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".
 */
void e_led_clear(void)
{
LED0 = 0;
LED1 = 0;
LED2 = 0;
```

```
LED3 = 0;
LED4 = 0;
LED5 = 0;
LED6 = 0;
LED7 = 0;
}

/*! \brief 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.
 * \param value 0 (off), 1 (on) otherwise change the state
 */
void e_set_body_led(unsigned int value)
{
if(value>1)
BODY_LED = BODY_LED^1;
else
BODY_LED = value;
}

/*! \brief 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.
 * \param value 0 (off), 1 (on) otherwise change the state
 */
void e_set_front_led(unsigned int value)
{
if(value>1)
FRONT_LED = FRONT_LED^1;
else
FRONT_LED = value;
}

/** \brief Change the state of all LED
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led(void)
{
LED0 = ~LED0;
LED1 = ~LED1;
LED2 = ~LED2;
LED3 = ~LED3;
LED4 = ~LED4;
LED5 = ~LED5;
LED6 = ~LED6;
LED7 = ~LED7;
}

/*! \brief Change the state of LED0
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led0(void)
{
LED0 = ~LED0;
}

/*! \brief Change the state of LED1
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led1(void)
{
LED1 = ~LED1;
}

/*! \brief Change the state of LED2
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led2(void)
{
LED2 = ~LED2;
}

/*! \brief Change the state of LED3
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led3(void)
{
LED3 = ~LED3;
}

/*! \brief Change the state of LED4
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led4(void)
{
LED4 = ~LED4;
}

/*! \brief Change the state of LED5
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
```

```
void e_blink_led5(void)
{
LED5 = ~LED5;
}

/*! \brief Change the state of LED6
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led6(void)
{
LED6 = ~LED6;
}

/*! \brief Change the state of LED7
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led7(void)
{
LED7 = ~LED7;
}

/*! \brief Start blinking all LED
 *
 * \param cycle    the number of cycle we wait before launching \ref e_blink_led(void)
 * \sa e_blink_led, e_activate_agenda
 */
void e_start_led_blinking(int cycle)
{
  e_activate_agenda(e_blink_led, cycle);
}

/*! \brief Stop blinking all LED
 *
 * This function use \ref e_destroy_agenda(void (*func)(void))
 * \sa e_destroy_agenda
 */
void e_stop_led_blinking(void)
{
  e_destroy_agenda(e_blink_led);
}

/*! \brief Change the blinking speed
 *
 * This function use \ref e_set_agenda_cycle(void (*func)(void), int cycle)
 * \param cycle    the number of cycle we wait before launching \ref e_blink_led(void)"
 * \sa e_blink_led, e_set_agenda_cycle
 */
void e_set_blinking_cycle(int cycle)
{
if (cycle>=0)
e_set_agenda_cycle(e_blink_led, cycle);
}

//#####################################################
// Artistics led effects
//#####################################################

#ifdef LED_EFFECTS

/*! \brief One led is on and turn clockwise */
void snake_led(void)
{
static unsigned char no_led = 0;
if(no_led == 0)
{
e_set_led(7, 0);
e_set_led(no_led, 1);
no_led++;
}
else if(no_led == 7)
{
e_set_led(no_led-1, 0);
e_set_led(no_led, 1);
no_led = 0;
}
else
{
e_set_led(no_led-1,0);
e_set_led(no_led, 1);
no_led++;
}
}

/*! \brief The leds go on from the front to the back
 and go off from the front to the back, etc */
void flow_led(void)
{
static unsigned char no_led = 0;
switch(no_led)
{
case 0: e_set_led(0, 2); break;
case 1: e_set_led(1, 2); e_set_led(7, 2); break;
case 2: e_set_led(2, 2); e_set_led(6, 2); break;
case 3: e_set_led(3, 2); e_set_led(5, 2); break;
case 4: e_set_led(4, 2); break;
}
if(no_led < 4)
no_led++;
else
no_led = 0;
}

/*! \brief The K2000 effect */
```

```
void k2000_led(void)
{
static unsigned char no_led = 0;
static unsigned char right = 1;
if(no_led == 0 && right) {
no_led = 1; e_set_led(0, 0); e_set_led(no_led, 1); right = 0;
}
else if(no_led == 0 && !right) {
no_led = 7; e_set_led(0, 0); e_set_led(no_led, 1); right = 1;
}
else if(no_led == 7) {
no_led = 0; e_set_led(7, 0); e_set_led(no_led, 1);
}
else if(no_led == 1) {
no_led = 0; e_set_led(1, 0); e_set_led(no_led, 1);
}
}

/*! \brief The right LED are indicating the right side */
void right_led(void)
{
static unsigned char no_led = 0;
switch(no_led)
{
case 0: e_set_led(0, 2); e_set_led(4, 2); break;
case 1: e_set_led(1, 2); e_set_led(3, 2); break;
case 2: e_set_led(2, 2); break;
case 3: e_led_clear();
}
if(no_led < 3)
no_led++;
else
no_led = 0;
}

/*! \brief The left LED are indicating the left side */
void left_led(void)
{
static unsigned char no_led = 0;
switch(no_led)
{
case 0: e_set_led(0, 2); e_set_led(4, 2); break;
case 1: e_set_led(7, 2); e_set_led(5, 2); break;
case 2: e_set_led(6, 2); break;
case 3: e_led_clear();
}
if(no_led < 3)
no_led++;
else
no_led = 0;
}
#endif
```

2.13.4 e_motors_NU.{h,c}

```
/*****************************************************************************

Advance control motor of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada

Modified Dec 2007, Kevin Lynch

This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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)
 * \code
 * #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) {}
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef _MOTORS
#define _MOTORS
```

```
/* internal functions */
//void run_left_motor(void);
//void run_right_motor(void);

/* user called function */
void e_init_motors(void);  // init to be done before using the other calls

void e_set_speed_left(int motor_speed);  // motor speed: from -1000 to 1000
void e_set_speed_right(int motor_speed); // motor speed: from -1000 to 1000
void e_set_speed(int linear_speed, int angular_speed);

void e_set_steps_left(int steps_left);
void e_set_steps_right(int steps_right);

int e_get_steps_left();
int e_get_steps_right();

/* below added by kml */

#define PI 3.1415926
#define WHEELRADIUS 2.06  // in cm, approximately
#define WHEELBASE 2.67  // half the distance between the wheels, approximately
#define STEPS_PER_REV 1000  // stepper motor steps per wheel revolution
#define MAXWHEELSPEED 2.0*PI // in rad/sec, based on max of 1 step every 2 interrupts, 0.5 ms between interrupts, 1000 steps per rev
#define MAXLINVEL MAXWHEELSPEED*WHEELRADIUS  // max lin vel of robot in cm/s
#define MAXANGVEL MAXLINVEL/WHEELBASE  // max ang vel about center in rad/s
#define LINSTEP PI*WHEELRADIUS/STEPS_PER_REV  // distance (in cm) traveled by center when one wheel ticks one step
#define ANGSTEP LINSTEP/WHEELBASE  // angle (rad) from one wheel tick
#define RAD2DEG 57.29578
#define DEG2RAD 1.0/RAD2DEG

void e_goal_active_left(int val);
void e_goal_active_right(int val);
void e_goal_steps_left(int stepsl);
void e_goal_steps_right(int stepsr);
int e_get_goal_active_left(void);
int e_get_goal_active_right(void);

float e_translate(float trans, float speed);  // in cm; positive trans = forward; returns speed in cm/s
float e_rotate(float ang, float speed);  // in radians; returns speed in rad/s
void e_get_configuration(float *xptr, float *yptr, float *thetaptr);  // returns x, y, theta
void e_set_configuration(float x, float y, float theta);  // sets current x, y, theta

#endif
```

```
/*****************************************************************************

Advance control motor of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada

Modified Dec 2007, Kevin Lynch, to add several functions.

This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the motors (with timer2)
 *
 * This module manage the motors with the agenda solution (timer2).
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include <math.h>
#include "e_motors_NU.h"
#include "./../e_epuck_ports.h"
#include "e_agenda_NU.h"
#include <stdlib.h>

/* If powersave is enabled, the motor library will not leave
 * the motor's phase powered when running at low speed, thus reducing the heat
 * dissipation and power consumption
 *
 * TRESHV == the "virtual" speed when powersave is enabled, the phase will
 * be keept on 1/TRESHV seconds.
 *
 * Powersave will only be enabled for speed < MAXV
 */

/* kml:  disabled POWERSAVE feature which had values chosen
 *       for 100 microsecond interrupts, now set to 500 microseconds.
 */

//#define POWERSAVE
#define TRESHV 650
#define MAXV 601

/*
CYCLE_BASE was 10000 in original version.
A motor moves one tick when CYCLE_BASE/motor_speed interrupts have
gone by.  But the original e_agenda.c used 100 microseconds between
```

```
interrupts, which we changed to 500 microseconds to allow more
to be done during an interrupt service routine (particularly to
allow cosines and sines for dead reckoning), so to keep the same
maximum speed of the motors, we lowered CYCLE_BASE from 10000 to
2000.  At max speed of 1000, the motor ticks every 2 interrupts.
*/
#define CYCLE_BASE 2000

#if TRESHV <= MAXV
#error TRESHV must be higher than MAXV
#endif

/* internal variables */

/* kml:  added below */

static int goal_active_left = 0;     // 1 if left motor is moving to a goal position
static int goal_active_right = 0;    // 1 if right motor is moving to a goal position
static int goal_steps_left = 0;      // the left motor goal position
static int goal_steps_right = 0;     // the right motor goal position
static float xpos = 0.0;             // the robot's current x position
static float ypos = 0.0;             // the robot's current y position
static float thetapos = 0.0;         // the robot's current angle
static float cval,sval;

/* kml:  added above */

static int left_speed = 0;           // speed of left motor, -1000 to 1000
static int right_speed = 0;          // speed of right motor, -1000 to 1000
static int left_motor_phase=0;      // phase can be 0 to 3
static int right_motor_phase=0;      // phase can be 0 to 3
static int nbr_steps_left=0;         // the number of steps of left motor since zeroed
static int nbr_steps_right=0;        // the number of steps of right motor since zeroed

/*------ internal calls ------*/

// returns sign of a float
int mysign(float val)
{
  if (val>0) return 1;
  else if (val<0) return -1;
  return 0;
}

/*! Change left motor phase according to the left_speed sign. */
 void run_left_motor(void)  // interrupt for motor 1 (of two) = left motor
{
  // increment or decrement phase depending on direction

#ifdef POWERSAVE
  static int phase_on = 0;
  if(phase_on && abs(left_speed) < MAXV) {
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
    MOTOR1_PHC = 0;
    MOTOR1_PHD = 0;
    phase_on = 0;
    e_set_agenda_cycle(run_left_motor, CYCLE_BASE/abs(left_speed) - CYCLE_BASE/TRESHV);
    return;
  }
#endif

  cval = cos(thetapos);
  sval = sin(thetapos);

  if (left_speed > 0) // inverted for the two motors
  {
    nbr_steps_left++;
    left_motor_phase--;
    if (left_motor_phase < 0) left_motor_phase = 3;

    // update robot's configuration estimate
    xpos = xpos + cval*LINSTEP;
    ypos = ypos + sval*LINSTEP;
thetapos = thetapos - ANGSTEP;
    if (thetapos<-PI) thetapos += 2*PI;
  }
  else
  {
    nbr_steps_left--;
    left_motor_phase++;
    if (left_motor_phase > 3) left_motor_phase = 0;

    // update robot's configuration estimate
    xpos = xpos - cval*LINSTEP;
    ypos = ypos - sval*LINSTEP;
thetapos = thetapos + ANGSTEP;
    if (thetapos>PI) thetapos -= 2*PI;
  }

  // set the phase on the port pins

  switch (left_motor_phase)
  {
    case 0:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 1;
```

```
      MOTOR1_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
  }
#ifdef POWERSAVE
  if(abs(left_speed) < MAXV) {
    phase_on = 1;
    e_set_agenda_cycle(run_left_motor,CYCLE_BASE/TRESHV);
  }
#endif

  // added below by kml
  if ((goal_active_left==1) && (nbr_steps_left == goal_steps_left)) {
    goal_active_left = 0;
    left_speed = 0;
    e_set_agenda_cycle(run_left_motor,0);
  }
}


/*! Change right motor phase according to the right_speed sign */
void run_right_motor(void)  // interrupt for motor 2 (of two) = right motor
{
  // increment or decrement phase depending on direction
#ifdef POWERSAVE
  static int phase_on = 0;
  if(phase_on && abs(right_speed) < MAXV) {
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
MOTOR2_PHC = 0;
MOTOR2_PHD = 0;
phase_on = 0;
e_set_agenda_cycle(run_right_motor, CYCLE_BASE/abs(right_speed) - CYCLE_BASE/TRESHV);
return;
  }
#endif

  cval = cos(thetapos);
  sval = sin(thetapos);

  if (right_speed < 0)
  {
nbr_steps_right--;
    right_motor_phase--;
    if (right_motor_phase < 0) right_motor_phase = 3;

    // update robot's configuration estimate
    xpos = xpos - cval*LINSTEP;
    ypos = ypos - sval*LINSTEP;
thetapos = thetapos - ANGSTEP;
    if (thetapos<-PI) thetapos += 2*PI;
  }
  else
  {
    nbr_steps_right++;
    right_motor_phase++;
    if (right_motor_phase > 3) right_motor_phase = 0;

    // update robot's configuration estimate
    xpos = xpos + cval*LINSTEP;
    ypos = ypos + sval*LINSTEP;
thetapos = thetapos + ANGSTEP;
    if (thetapos>PI) thetapos -= 2*PI;
  }

  // set the phase on the port pins

  switch (right_motor_phase)
  {
    case 0:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
```

```
      break;
    }
    case 3:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
  }
#ifdef POWERSAVE
  if(abs(right_speed) < MAXV) {
    phase_on = 1;
    e_set_agenda_cycle(run_right_motor,CYCLE_BASE/TRESHV);
  }
#endif

  /* added below by kml */
  if ((goal_active_right==1) && (nbr_steps_right == goal_steps_right)) {
    goal_active_right = 0;
    right_speed = 0;
    e_set_agenda_cycle(run_right_motor,0);
  }
}

/* ---- user calls ---- */

/*! \brief Initialize the motors's agendas
 *
 * This function initialize the agendas used by the motors. In fact
 * it call \ref e_activate_agenda(void (*func)(void), int cycle) function.
 * \sa e_activate_agenda
 */
void e_init_motors(void)
{
  e_activate_agenda(run_left_motor, 0);
  e_activate_agenda(run_right_motor, 0);
}

/*! \brief 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 \ref e_set_agenda_cycle(void (*func)(void), int cycle)).
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 * \sa e_set_agenda_cycle
 */
void e_set_speed_left(int motor_speed)
{
  // speed null
  if (motor_speed == 0)
  {
    left_speed = 0;
    e_set_agenda_cycle(run_left_motor, 0);
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
    MOTOR1_PHC = 0;
    MOTOR1_PHD = 0;
  }
  // speed inferior to the minimum value
  else if(motor_speed < -1000)
  {
    left_speed = -1000;
    e_set_agenda_cycle(run_left_motor, (int)-CYCLE_BASE/left_speed);
  }
  // speed superior to the maximum value
  else if(motor_speed > 1000)
  {
    left_speed = 1000;
    e_set_agenda_cycle(run_left_motor, (int) CYCLE_BASE/left_speed);
  }
  else
  {
    left_speed = motor_speed;
  // negative speed
if(motor_speed < 0)
e_set_agenda_cycle(run_left_motor, (int)-CYCLE_BASE/motor_speed);
  // positive speed
else
e_set_agenda_cycle(run_left_motor, (int) CYCLE_BASE/motor_speed);
  }
}

/*! \brief 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 \ref e_set_agenda_cycle(void (*func)(void), int cycle)).
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 * \sa e_set_agenda_cycle
 */
void e_set_speed_right(int motor_speed)  // motor speed in percent
{
  // speed null
  if (motor_speed == 0)
  {
    right_speed = 0;
    e_set_agenda_cycle(run_right_motor, 0);
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
    MOTOR2_PHC = 0;
    MOTOR2_PHD = 0;
  }
  // speed inferior to the minimum value
```

```
  else if (motor_speed < -1000)
  {
    right_speed = -1000;
    e_set_agenda_cycle(run_right_motor, (int)-CYCLE_BASE/right_speed);
  }
  // speed superior to the maximum value
  else if (motor_speed > 1000)
  {
    right_speed = 1000;
    e_set_agenda_cycle(run_right_motor, (int) CYCLE_BASE/right_speed);
  }
  else
  {
    right_speed = motor_speed;

  // negative speed
if(motor_speed < 0)
e_set_agenda_cycle(run_right_motor, (int)-CYCLE_BASE/motor_speed);
  // positive speed
else
e_set_agenda_cycle(run_right_motor, (int) CYCLE_BASE/motor_speed);
  }
}

/*! \brief Manage linear/angular speed
 *
 * This function manage the speed of the motors according to the
 * desired linear and angular speed.
 * \param linear_speed the speed in the axis of e-puck
 * \param angular_speed the rotation speed (trigonometric)
 */
void e_set_speed(int linear_speed, int angular_speed)
{
if(abs(linear_speed) + abs(angular_speed) > 1000)
return;
else
{
e_set_speed_left (linear_speed - angular_speed);
e_set_speed_right(linear_speed + angular_speed);
}
}

/*! \brief Give the number of left motor steps
 * \return The number of phases steps made since the left motor
 * is running.
 */
int e_get_steps_left()
{
return nbr_steps_left;
}

/*! \brief Set the number of left motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_left(int set_steps)
{
nbr_steps_left=set_steps;
}

/*! \brief Give the number of right motor steps
 * \return The number of phases steps made since the right motor
 * is running.
 */
int e_get_steps_right()
{
return nbr_steps_right;
}

/*! \brief Set the number of right motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_right(int set_steps)
{
nbr_steps_right=set_steps;
}

/****** below added by kml *******/

// if 0, goal becomes inactive; if 1, goal is active
void e_goal_active_left(int val)
{
  if (val==0)
    goal_active_left = 0;
  else goal_active_left = 1;
}

// if 0, goal becomes inactive; if 1, goal is active
void e_goal_active_right(int val)
{
  if (val==0)
    goal_active_right = 0;
  else goal_active_right = 1;
}

// set the number of steps to get to the goal
void e_goal_steps_left(int stepsl)
{
  goal_steps_left = stepsl;
}

// set the number of steps to get to the goal
void e_goal_steps_right(int stepsr)
{
  goal_steps_right = stepsr;
}

// return whether a goal is active or not
```

```
int e_get_goal_active_left(void)
{
  return goal_active_left;
}

// return whether a goal is active or not
int e_get_goal_active_right(void)
{
  return goal_active_right;
}

/*
Robot translates "trans" cm at "speed" cm/s, where a negative value
makes it go backwards.
*/
float e_translate(float trans, float speed)
{
  int goalsteps,intspeed;

  speed = abs(speed);
  if (speed>MAXLINVEL) speed = MAXLINVEL;
  if (speed==0) return speed;
  goalsteps = (int) (0.5*trans/((float) (LINSTEP)));
  if (goalsteps==0) return 0;
  intspeed = (int) (speed*1000.0/((float) (MAXLINVEL)));
  intspeed = intspeed*mysign(trans);

  e_set_steps_left(0);
  e_set_steps_right(0);
  e_goal_steps_left(goalsteps);
  e_goal_steps_right(goalsteps);
  e_set_speed_left(intspeed);
  e_set_speed_right(intspeed);
  e_goal_active_left(1);
  e_goal_active_right(1);
  return speed;
}

/*
Robot rotates "ang" radians at "speed" radians per second.
*/
float e_rotate(float ang, float speed)
{
  int goalsteps,intspeed,thesign;

  speed = abs(speed);
  if (speed>MAXANGVEL) speed = MAXANGVEL;
  if (speed==0) return speed;
  thesign = mysign(ang);
  goalsteps = (int) (0.5*ang/((float) (ANGSTEP)));
  if (goalsteps==0) return 0;
  intspeed = (int) (speed*1000/((float) (MAXANGVEL)));

  e_set_steps_left(0);
  e_set_steps_right(0);
  e_goal_steps_left(-1*goalsteps);
  e_goal_steps_right(goalsteps);
  e_set_speed_left(-1*intspeed*thesign);
  e_set_speed_right(intspeed*thesign);
  e_goal_active_left(1);
  e_goal_active_right(1);
  return speed;
}

// return the configuration of the robot
void e_get_configuration(float *xptr, float *yptr, float *thetaptr)
{
  *xptr = xpos;
  *yptr = ypos;
  *thetaptr = thetapos;
}

// set the configuration of the robot (e.g., due to new sensory information)
void e_set_configuration(float x, float y, float theta)
{
  xpos = x;
  ypos = y;
  thetapos = theta;
}
```

2.13.5 e_motors.{h,c}

```
/*****************************************************************************

Advance control motor of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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)
 * \code
 * #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) {}
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef _MOTORS
#define _MOTORS

/* internal functions */
//void run_left_motor(void);
//void run_right_motor(void);

/* user called function */
void e_init_motors(void);  // init to be done before using the other calls

void e_set_speed_left(int motor_speed);  // motor speed: from -1000 to 1000
void e_set_speed_right(int motor_speed); // motor speed: from -1000 to 1000
void e_set_speed(int linear_speed, int angular_speed);

void e_set_steps_left(int steps_left);
void e_set_steps_right(int steps_right);

int e_get_steps_left();
int e_get_steps_right();

#endif
```

```
/*****************************************************************************

Advance control motor of e-puck
December 2004: first version
Lucas Meier & Francesco Mondada


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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)
 * \code
 * #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) {}
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include "./../e_epuck_ports.h"
#include "e_agenda.h"
#include <stdlib.h>

/* If powersave is enabled, the motor library will not leave
 * the motor's phase powered when running at low speed, thus reducing the heat
 * dissipation and power consumption
 *
 * TRESHV == the "virtual" speed when powersave is enabled, the phase will
 * be keept on 1/TRESHV seconds.
 *
 * Powersave will only be enabled for speed < MAXV
```

```
 */
#define POWERSAVE
#define TRESHV 650
#define MAXV 601

#if TRESHV <= MAXV
#error TRESHV must be higher than MAXV
#endif

/* internal variables */
static int left_speed = 0;
static int right_speed = 0;

static int left_motor_phase=0;     // phase can be 0 to 3
static int right_motor_phase=0;    // phase can be 0 to 3

static int nbr_steps_left=0;
static int nbr_steps_right=0;

/*------ internal calls ------*/

/*! Change left motor phase according to the left_speed sign. */
 void run_left_motor(void)  // interrupt for motor 1 (of two) = left motor
{
  // increment or decrement phase depending on direction

#ifdef POWERSAVE
  static int phase_on = 0;
  if(phase_on && abs(left_speed) < MAXV) {
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
MOTOR1_PHC = 0;
MOTOR1_PHD = 0;
phase_on = 0;
e_set_agenda_cycle(run_left_motor, 10000/abs(left_speed) - 10000/TRESHV);
return;
  }
#endif

  if (left_speed > 0) // inverted for the two motors
  {
    nbr_steps_left++;
    left_motor_phase--;
    if (left_motor_phase < 0) left_motor_phase = 3;
  }
  else
  {
    nbr_steps_left--;
    left_motor_phase++;
    if (left_motor_phase > 3) left_motor_phase = 0;
  }

  // set the phase on the port pins

  switch (left_motor_phase)
  {
    case 0:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
  }
#ifdef POWERSAVE
  if(abs(left_speed) < MAXV) {
    phase_on = 1;
    e_set_agenda_cycle(run_left_motor,10000/TRESHV);
  }
#endif
}

/*! Change right motor phase according to the right_speed sign */
void run_right_motor(void)  // interrupt for motor 2 (of two) = right motor
{
  // increment or decrement phase depending on direction
#ifdef POWERSAVE
  static int phase_on = 0;
  if(phase_on && abs(right_speed) < MAXV) {
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
MOTOR2_PHC = 0;
```

```
MOTOR2_PHD = 0;
phase_on = 0;
e_set_agenda_cycle(run_right_motor, 10000/abs(right_speed) - 10000/TRESHV);
return;
  }
#endif
  if (right_speed < 0)
  {
nbr_steps_right--;
    right_motor_phase--;
    if (right_motor_phase < 0) right_motor_phase = 3;
  }
  else
  {
    nbr_steps_right++;
    right_motor_phase++;
    if (right_motor_phase > 3) right_motor_phase = 0;
  }

  // set the phase on the port pins

  switch (right_motor_phase)
  {
    case 0:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
  }
#ifdef POWERSAVE
  if(abs(right_speed) < MAXV) {
    phase_on = 1;
    e_set_agenda_cycle(run_right_motor,10000/TRESHV);
  }
#endif
}

/* ---- user calls ---- */

/*! \brief Initialize the motors's agendas
 *
 * This function initialize the agendas used by the motors. In fact
 * it call \ref e_activate_agenda(void (*func)(void), int cycle) function.
 * \sa e_activate_agenda
 */
void e_init_motors(void)
{
  e_activate_agenda(run_left_motor, 0);
  e_activate_agenda(run_right_motor, 0);
}

/*! \brief 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 \ref e_set_agenda_cycle(void (*func)(void), int cycle)).
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 * \sa e_set_agenda_cycle
 */
void e_set_speed_left(int motor_speed)
{
  // speed null
  if (motor_speed == 0)
  {
    left_speed = 0;
    e_set_agenda_cycle(run_left_motor, 0);
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
    MOTOR1_PHC = 0;
    MOTOR1_PHD = 0;
  }
  // speed inferior to the minimum value
  else if(motor_speed < -1000)
  {
    left_speed = -1000;
    e_set_agenda_cycle(run_left_motor, (int)-10000/left_speed);
  }
  // speed superior to the maximum value
  else if(motor_speed > 1000)
```

```
  {
    left_speed = 1000;
    e_set_agenda_cycle(run_left_motor, (int) 10000/left_speed);
  }
  else
  {
    left_speed = motor_speed;
  // negative speed
if(motor_speed < 0)
e_set_agenda_cycle(run_left_motor, (int)-10000/motor_speed);
  // positive speed
else
e_set_agenda_cycle(run_left_motor, (int) 10000/motor_speed);

  }
}

/*! \brief 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 \ref e_set_agenda_cycle(void (*func)(void), int cycle)).
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 * \sa e_set_agenda_cycle
 */
void e_set_speed_right(int motor_speed)  // motor speed in percent
{
  // speed null
  if (motor_speed == 0)
  {
    right_speed = 0;
    e_set_agenda_cycle(run_right_motor, 0);
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
    MOTOR2_PHC = 0;
    MOTOR2_PHD = 0;
  }
  // speed inferior to the minimum value
  else if (motor_speed < -1000)
  {
    right_speed = -1000;
    e_set_agenda_cycle(run_right_motor, (int)-10000/right_speed);
  }
  // speed superior to the maximum value
  else if (motor_speed > 1000)
  {
    right_speed = 1000;
    e_set_agenda_cycle(run_right_motor, (int) 10000/right_speed);
  }
  else
  {
    right_speed = motor_speed;

  // negative speed
if(motor_speed < 0)
e_set_agenda_cycle(run_right_motor, (int)-10000/motor_speed);
  // positive speed
else
e_set_agenda_cycle(run_right_motor, (int) 10000/motor_speed);

  }
}

/*! \brief Manage linear/angular speed
 *
 * This function manage the speed of the motors according to the
 * desired linear and angular speed.
 * \param linear_speed the speed in the axis of e-puck
 * \param angular_speed the rotation speed (trigonometric)
 */
void e_set_speed(int linear_speed, int angular_speed)
{
if(abs(linear_speed) + abs(angular_speed) > 1000)
return;
else
{
e_set_speed_left (linear_speed - angular_speed);
e_set_speed_right(linear_speed + angular_speed);
}
}

/*! \brief Give the number of left motor steps
 * \return The number of phases steps made since the left motor
 * is running.
 */
int e_get_steps_left()
{
return nbr_steps_left;
}

/*! \brief Set the number of left motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_left(int set_steps)
{
nbr_steps_left=set_steps;
}

/*! \brief Give the number of right motor steps
 * \return The number of phases steps made since the right motor
 * is running.
 */
int e_get_steps_right()
{
return nbr_steps_right;
}
```

```
/*! \brief Set the number of right motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_right(int set_steps)
{
nbr_steps_right=set_steps;
}
```

2.13.6 e_remote_control.{h,c}

```
/*****************************************************************************

control IR receiver module
september 2005 : first version
Valentin Longchamp


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Valentin Longchamp

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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)
 * \code
 * #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) {}
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Valentin Longchamp \n Doc: Jonathan Besuchet
 */

/*****************************************************************
*  control IR receiver module *
* september 2005 : first version *
* Valentin Longchamp *
* *
*****************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the IR receiver module (timer2)
 *
 * 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).
 * \code
 * #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;
 *  }
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Jonathan Besuchet
 */

#ifndef _IR_REMOTE_CONTROLE
#define _IR_REMOTE_CONTROLE

/* defines for the keys on the remote controler
 * the numbers 0 to 9 are not defined since they are the same */
#define BOTTOMR 10  // -/-- key
#define BOTTOM1 11  // p<p key
```

```
#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);
void e_read_remote_control(void);

unsigned char e_get_check(void);
unsigned char e_get_address(void);
unsigned char e_get_data(void);

#endif
```

```
/*****************************************************************************

Control IR receiver module
December 2005: first version
Valentin Longchamp


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Valentin Longchamp

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch
*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief 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).
 * \code
 * #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;
 *  }
 * }
 * \endcode
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include "e_remote_control.h"
#include "../e_epuck_ports.h"
#include "e_agenda.h"
#include "e_led.h"

/*------ internal 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;


/*! \brief Initialise the IR receiver ports */
void e_init_remote_control(void) // initialisation for IR interruptions on INT0
{
REMOTE_DIR = INPUT_PIN; //sets the RF6 pin as input
INTCON2bits.INT0EP = 1;    //set interrupt polarity to falling edge
IFS0bits.INT0IF = 0; //clear to enable interrupt
IEC0bits.INT0IE = 1; //enable interrupt on INT0
return;
}

void __attribute__((__interrupt__, auto_psv))
 _INT0Interrupt(void) // interrupt for IR receiver
{
```

```
IEC0bits.INT0IE = 0;    //disable interrup from falling edge
// e_set_led(1,1);
e_activate_agenda(e_read_remote_control, 21); //activate the IR Receiver agenda with a 2.1[ms] cycle value
check_temp = address_temp = data_temp = 0;
return;
}

/*! \brief Read the signal and stock the information */
void e_read_remote_control(void) // interrupt from timer for next bits
{
static int i = -1;


if (i == -1) // start bit confirm  change timer period
{
if(REMOTE){
//if high it is only a noise
IEC0bits.INT0IE = 1;    //enable interrupt from falling edge
IFS0bits.INT0IF = 0;    //clear interrupt flag from first receive !
e_destroy_agenda(e_read_remote_control);
i = -1;
}
else    // read the check bit
{
e_set_agenda_cycle(e_read_remote_control, 9); //cycle value is 0.9 to go to check bit[ms]
check_temp = address_temp = data_temp = 0;
//e_set_led(1,1);
i=0;
}
}
// e_set_led(2,1);

else if (i == 1) // check bit read and change timer period
{
// e_set_led(3,1);
check_temp = REMOTE;    // read the check bit
e_set_agenda_cycle(e_read_remote_control, 18); //cycle value is 1.778[ms]
e_set_led(1,1);
}
else if ((i > 1) && (i < 7)) // we read address
{
// e_set_led(4,1);

unsigned char temp = REMOTE;
temp <<= 6-i;
address_temp += temp;
}
else if ((i > 6) && (i < 13 )) // we read data
{
// e_set_led(5,1);

unsigned char temp = REMOTE;
temp <<= 6+6-i;
data_temp += temp;
}

else if (i == 13) // last bit read
{
e_set_led(1,0);
IEC0bits.INT0IE = 1;    //enable interrupt from falling edge
IFS0bits.INT0IF = 0;    //clear interrupt flag from first receive !
e_destroy_agenda(e_read_remote_control);
i = -1;
check = check_temp;
address = address_temp;
data = data_temp;
}

if(i!=-1)
i++;
}

/*------ user calls ------*/

/** \brief Read the check bit
 * \return check check bit of the signal
 */
unsigned char e_get_check(void) {
return check;
}

/** \brief Read the adress of the commande
 * \return adress adress part of the signal
 */
unsigned char e_get_address(void) {
return address;
}

/** \brief Read the data of the command
 * \return data data part of the signal
 */
unsigned char e_get_data(void) {
return data;
}
```

2.14 motor_led/advance_one_timer/fast_agenda

2.14.1 e_agenda_fast.{h,c}

```
/*****************************************************************************

Advance fast agenda events of e-puck
2005: first version
```

```
Julien Hubert


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Julien Hubert

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

**********************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the fast agendas (timer1, 2, 3).
 *
 * This module manage the fast agendas with the timer1, 2, 3.
 * \n \n 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.
 * \n \n 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.
 * \n \n If one (or more) agenda has to be treated, his callback function is launch.
 * \author Code: Julien Hubert \n Doc: Jonathan Besuchet
 */

#ifndef __AGENDA_FAST_H__
#define __AGENDA_FAST_H__

#define AG_ALREADY_CREATED 1
#define AG_NOT_FOUND 2

/*****************************************************************
 * ------------------------ Type definition -----------------------
 *****************************************************************/

typedef struct AgendaType Agenda;

struct AgendaType
{
  unsigned int cycle; // length in 10e of ms of a cycle between two alarms
  int  counter; // counter
  char activate; // can be on=1 or off=0
  void (*function) (void); // function called when counter > cycle,
// WARNING: This function must have the following prototype
// void func(void)
  Agenda *next; // pointer on the next agenda
};

/*! \struct AgendaList
 * \brief 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.
 */
struct AgendaList
{
char motors;
Agenda* waiting;                  /*!< If an agenda's speed goes down to zero, we remove it from the list and add it to the waiting list */
Agenda* agendas[3];               /*!< We use the 3 first timers */
unsigned char timers_in_use[3];   /*!< Determine which one we use currently */
unsigned speed[3];                /*!< Base speed 0.1 ms but use of multiplication*/
};

/*****************************************************************
 * ------------------------ From agenda.c file -------------------------
 *****************************************************************/
void e_start_agendas_processing(void);
void e_start_timer_processing(int timer);

//void e_end_agendas_processing();
void e_end_agendas_processing(int timer);

void e_configure_timer(int timer);

void e_activate_agenda(void (*func)(void), unsigned cycle);
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));

int e_set_agenda_cycle(void (*func)(void), unsigned cycle);
int e_reset_agenda(void (*func)(void));

int e_pause_agenda(void (*func)(void));
int e_restart_agenda(void (*func)(void));

//void SendAgendaStatus();


#endif /* __AGENDA_FAST_H__ */


/* End of File : e_agenda_fast.h */
```

```
/**********************************************************************************

Advance fast agenda events of e-puck
2005: first version
```

```
Julien Hubert


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Julien Hubert

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the fast agendas (timer1, 2, 3).
 *
 * This module manage the fast agendas with the timer1, 2, 3.
 * \n \n 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.
 * \n \n 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.
 * \n \n If one (or more) agenda has to be treated, his callback function is launch.
 * \author Code: Julien Hubert \n Doc: Jonathan Besuchet
 */

#include "e_agenda_fast.h"

#include "../../e_epuck_ports.h"
#include <stdlib.h>

#define EXIT_OK 1

//static char buffer[100];

static struct AgendaList agenda_list;

// Use Timer 1, 2 and 3
// With prescaler == 0 we can do only 4.4 msec timers at maximum.
// So we need to use the prescaler
// prescaler == 1 -> 1:8 prescaler -> up to 35.8 msec
// prescaler == 2 -> 1:64 prescaler -> up to 286.4 msec
// prescaler == 3 -> 1:256 prescaler -> up to 1145.9 msec

/*------ internal calls ------*/

// Copyright Wikipedia A faster one in assembly language and binary operators can be found
unsigned compute_gcd(unsigned u,unsigned v)
{
     unsigned int k = 0;
     if (u == 0)
         return v;
     if (v == 0)
         return u;
     while ((u & 1) == 0  &&  (v & 1) == 0) { /* while both u and v are even */
         u >>= 1;   /* shift u right, dividing it by 2 */
         v >>= 1;   /* shift v right, dividing it by 2 */
         k++;       /* add a power of 2 to the final result */
     }
     /* At this point either u or v (or both) is odd */
     do {
         if ((u & 1) == 0)      /* if u is even */
             u >>= 1;           /* divide u by 2 */
         else if ((v & 1) == 0) /* else if v is even */
             v >>= 1;           /* divide v by 2 */
         else if (u >= v)       /* u and v are both odd */
             u = (u-v) >> 1;
         else                   /* u and v both odd, v > u */
             v = (v-u) >> 1;
     } while (u > 0);
     return v << k;  /* returns v * 2^k */

}

unsigned my_ceil(float a)
{
if (a>(unsigned)a)
return (unsigned)a+1;
return (unsigned)a;
}

void e_set_timer_speed(int timer, unsigned speed)
{
// Compute adequate prescaler
unsigned prescaler=0;
unsigned factor=0;

if (speed>11459)
speed=speed/my_ceil(speed/11459.0);

agenda_list.speed[timer-1]=speed;

if (speed>2864) {
prescaler=3;
factor=8;   //2^8
}
else {
  if (speed>358) {
prescaler=2;
factor=6;   //2^6
}
else
if (speed>=44) {  // 44 = (2^16-1)/1464
```

```
prescaler=1;
factor=3;   //2^3
}
}

switch(timer) {
case 0:
e_set_timer_speed(1,speed);
e_set_timer_speed(2,speed);
e_set_timer_speed(3,speed);
break;
case 1:
PR1 = (unsigned)((long unsigned)(0.1*MILLISEC*speed) >> factor);
T1CONbits.TCKPS=prescaler;
break;
case 2:
PR2 = (unsigned)((long unsigned)(0.1*MILLISEC*speed) >> factor);
T2CONbits.TCKPS=prescaler;
break;
case 3:
PR3 = (unsigned)((long unsigned)(0.1*MILLISEC*speed) >> factor);
T3CONbits.TCKPS=prescaler;
break;
default:
break;
}
}

// Look for the best timer to receive a task with cycle
int search_best_fit(unsigned cycle)
{
int i=0;
char ctimer=-1;
unsigned gcd=1;
unsigned tgcd;
for (i=0;i<3;++i)
if (agenda_list.timers_in_use[i] > 0 && ((tgcd=compute_gcd(agenda_list.speed[i],cycle))>gcd) ) {
gcd=tgcd;
ctimer=i;
}
if (ctimer<0) {
ctimer=0;
gcd=cycle;
}
return ctimer;
}

// Recompute the speeds of all the timers to optimize their cycles
void recompute_speeds()
{
e_end_agendas_processing(0);   // Stops the agenda
unsigned gcd;
int i=1;
Agenda* current;

for (;i<3;++i) {
current=agenda_list.agendas[i];
if (current) {
gcd=current->cycle;
current=current->next;
}
while (current) {
gcd=compute_gcd(gcd,current->cycle);
current=current->next;
}
if (gcd!=agenda_list.speed[i]) {
e_set_timer_speed(i+1,gcd);
}
}
e_start_timer_processing(0);  // Resume the timers
}

// Find a particular function
Agenda* find_function(void (*func)(void))
{
Agenda *current=agenda_list.waiting;
int i;

while(current && current->function != func)
current = current->next;
if(current)
return current;
else {

for (i=0;i<3;++i) {
current = agenda_list.agendas[i];
while(current && current->function != func)
current = current->next;
if (current)
return current;
}
}
return (Agenda*)0;
}

// Assign a task to the best timer
unsigned assign_agenda(Agenda* agenda)
{
unsigned gcd=1,tgcd;
int i;
char ctimer=-1;

// Look for one free timer...

for (i=0;i<3 && agenda_list.timers_in_use[i] > 0;++i);

if (i<3 && agenda_list.motors!=i) {   // We have a free timer...
```

```
e_configure_timer(i+1);   // Initialise the timer
ctimer=i;
gcd=agenda->cycle;
}
else {
for (i=0;i<3;++i)
if (agenda_list.timers_in_use[i] > 0 && agenda_list.motors!=i && ((tgcd=compute_gcd(agenda_list.speed[i],agenda->cycle))>gcd) ) {
gcd=tgcd;
ctimer=i;
}
}

if (ctimer<0) {
ctimer=0;

if (ctimer==agenda_list.motors)
ctimer=1;
else if (agenda_list.motors!=1 && agenda_list.speed[1]<agenda_list.speed[(unsigned)ctimer])
ctimer=1;

if (agenda_list.motors!=2 && agenda_list.speed[2]<agenda_list.speed[(unsigned)ctimer])
ctimer=2;
}

e_end_agendas_processing(ctimer+1);
e_set_timer_speed(ctimer+1,gcd);

agenda->activate=1;
agenda->next=agenda_list.agendas[(unsigned)ctimer];
agenda_list.agendas[(unsigned)ctimer]=agenda;
agenda_list.timers_in_use[i]++;
e_start_timer_processing(ctimer+1);

return ctimer;
}

void migrate(int timer)
{
if (agenda_list.timers_in_use[timer]>0) {
Agenda* current=agenda_list.agendas[timer];
Agenda* next=0;

while (current) {
next=current->next;
current->next=0;
assign_agenda(current);
current=next;
}
agenda_list.agendas[timer]=0;
agenda_list.timers_in_use[timer]=0;
}
}

/*------ external calls ------*/

/*! \brief Initialise the accounting structure
 *
 * Start the agendas processing by initialising the accounting structures.
 * \n Don't activate any timer which is done by e_start_timer_processing.
 * \sa e_start_timer_processing
 */
void e_start_agendas_processing(void)
{
agenda_list.agendas[0] = 0;
agenda_list.agendas[1] = 0;
agenda_list.agendas[2] = 0;

agenda_list.speed[0] = 1;
agenda_list.speed[1] = 1;
agenda_list.speed[2] = 1;

agenda_list.waiting = 0;

agenda_list.timers_in_use[0] = 0;
agenda_list.timers_in_use[1] = 0;
agenda_list.timers_in_use[2] = 0;

agenda_list.motors=-1;
}


/*! \brief Configure the timer(s) used
 *
 * Configure one or all the timers to be used by the agenda
 * \param timer the timer's number to configure (between [0-3] with 0 configuring all of them)
 */
void e_configure_timer(int timer)
{
switch(timer) {
case 0:
e_configure_timer(1);
e_configure_timer(2);
e_configure_timer(3);
break;
case 1:
T1CON = 0; // reset Timer1 CONtrol register
T1CONbits.TCKPS = 0; // prescaler = 1
TMR1 = 0; // clear timer 1
PR1 = (int)(0.1*MILLISEC);     // interrupt every 0.1 ms without prescaler
IFS0bits.T1IF = 0; // clear interrupt flag
IEC0bits.T1IE = 1; // set interrupt enable bit
break;
case 2:
T2CON = 0; // reset Timer2 CONtrol register
T2CONbits.TCKPS = 0;          // prescaler = 1
TMR2 = 0; // clear timer 2
PR2 = (int)(0.1*MILLISEC); // interrupt every 0.1 ms without prescaler
```

```
IFS0bits.T2IF = 0; // clear interrupt flag
IEC0bits.T2IE = 1; // set interrupt enable bit
break;
case 3:
T3CON = 0; // reset Timer3 CONtrol register
T3CONbits.TCKPS = 0; // prescaler = 1
TMR3 = 0; // clear timer 3
PR3 = (int)(0.1*MILLISEC);     // interrupt every 0.1 ms without prescaler
IFS0bits.T3IF = 0; // clear interrupt flag
IEC0bits.T3IE = 1; // set interrupt enable bit
break;
default:
break;
}
}

/*! \brief Start the timer(s) used
 *
 * Activate one or all the timers to be used by the agenda.
 * \param timer the timer's number to activate (between [0-3] with 0 activating all of them)
 */
void e_start_timer_processing(int timer)
{
switch(timer) {
case 0:
e_start_timer_processing(1);
e_start_timer_processing(2);
e_start_timer_processing(3);
break;
case 1:
T1CONbits.TON = 1; // start Timer1
// agenda_list.timers_in_use[0] = 1;
break;
case 2:
T2CONbits.TON = 1; // start Timer2
// agenda_list.timers_in_use[1] = 1;
break;
case 3:
T3CONbits.TON = 1; // start Timer3
// agenda_list.timers_in_use[2] = 1;
break;
default:
break;
}
}

/*! \brief 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).
 * \param timer the timer's number [0-3]. 0 stops all the timers.
 * \sa e_destroy_agenda
 */
void e_end_agendas_processing(int timer)
{
switch(timer) {
case 0:
e_end_agendas_processing(1);
e_end_agendas_processing(2);
e_end_agendas_processing(3);
break;
case 1:
T1CONbits.TON = 0;
break;
case 2:
T2CONbits.TON = 0;
break;
case 3:
T3CONbits.TON = 0;
break;
default:
break;
}
}

/*! \brief 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).
 * \n The appropriate timer is automatically selected.
 * \param func function called if the cycle value is reached by the counter
 * \param cycle     cycle value in millisec/10
 */
void e_activate_agenda(void (*func)(void), unsigned cycle)
{
e_end_agendas_processing(0);

// Choose the best timer to put the task on
int i=0;
Agenda *current;
char ctimer=-1;
unsigned gcd=1;
unsigned tgcd;

if (cycle>0) {   // We don't insert the task if its cycle is zero (cfr motors)

// Look for a free timer
for (;i<3 && agenda_list.timers_in_use[i] > 0;++i);
if (i<3 && agenda_list.motors!=i) {   // We have at least one free timer so we'll use this one

e_configure_timer(i+1);   // Initialise the timer
e_set_timer_speed(i+1,cycle);
if (!(current = malloc(sizeof(Agenda))))
exit(EXIT_FAILURE);
agenda_list.agendas[i]=current;
current->cycle=cycle;
```

```
current->counter=0;
current->function=func;
current->activate=1;
current->next=0;
agenda_list.timers_in_use[i]++;
e_start_timer_processing(i+1);
}
else { // We have no free timer so we look for one that is the closest to the cycle we need
// First we compute the gcd with existing timers.

for (i=0;i<3;++i)
if (agenda_list.timers_in_use[i] > 0 && agenda_list.motors!=i && ((tgcd=compute_gcd(agenda_list.speed[i],cycle))>gcd) ) {
gcd=tgcd;
ctimer=i;
}


if (ctimer<0) {
ctimer=0;

if (agenda_list.speed[1]<agenda_list.speed[(unsigned)ctimer])
ctimer=1;

if (agenda_list.speed[2]<agenda_list.speed[(unsigned)ctimer])
ctimer=2;
}

// Now that we know where to add the new task, the new speed of the timer is in gcd
// 1. Stop the timer
// 2. Change its speed
// 3. Add the new task
// 4. Restart the timer

// e_end_agendas_processing(ctimer+1);
e_set_timer_speed(ctimer+1,gcd);

if (!(current = malloc(sizeof(Agenda))))
exit(EXIT_FAILURE);
current->cycle=cycle;
current->counter=0;
current->function=func;
current->activate=1;
current->next=agenda_list.agendas[(unsigned)ctimer];
agenda_list.agendas[(unsigned)ctimer]=current;
agenda_list.timers_in_use[(unsigned)ctimer]++;
// e_start_timer_processing((unsigned)ctimer+1);
}
}
else
{
// We create the structure of the new task and put it in the waiting list
if (!(current = malloc(sizeof(Agenda))))
exit(EXIT_FAILURE);
current->cycle=cycle;
current->counter=0;
current->function=func;
current->activate=0;
current->next=agenda_list.waiting;
agenda_list.waiting=current;
}
e_start_timer_processing(0);
}

void e_activate_motors(void (*func1)(void),void (*func2)(void))
{
// Find the less used timer to migrate. There is no reason to look at speed as the task will be spread on other timers

e_end_agendas_processing(0);

char i=0,timer=0;
unsigned nbtimer=65535;
Agenda* current;
for (;i<3 && agenda_list.timers_in_use[(int)i]>0;++i)
if (agenda_list.timers_in_use[(int)i]<nbtimer) {
nbtimer=agenda_list.timers_in_use[(int)i];
timer=i;
}

if (i==3) {    // We didn't find a free timer
agenda_list.motors=timer;
migrate(timer);
agenda_list.speed[(int)timer]=1;
}
else
agenda_list.motors=i;

e_configure_timer(agenda_list.motors+1);   // Initialise the timer
if (!(current = malloc(sizeof(Agenda))))
exit(EXIT_FAILURE);
agenda_list.agendas[(int)timer]=current;
current->cycle=0;
current->counter=0;
current->function=func1;
current->activate=1;
if (!(current->next=malloc(sizeof(Agenda))))
exit(EXIT_FAILURE);
current=current->next;
current->next=0;
current->cycle=0;
current->counter=0;
current->function=func2;
current->activate=1;

agenda_list.timers_in_use[(int)agenda_list.motors]+=2;

e_start_timer_processing(0);
}
```

```
int e_set_motor_speed(void (*func)(void), unsigned cycle)
{
if (agenda_list.motors>=0) {
Agenda *current=agenda_list.agendas[(int)agenda_list.motors];
while (current && current->function != func)
current=current->next;
if (!current)
return AG_NOT_FOUND;

e_end_agendas_processing(agenda_list.motors+1);
if (current->next)
e_set_timer_speed(agenda_list.motors+1,compute_gcd(current->next->cycle,cycle));
else
e_set_timer_speed(agenda_list.motors+1,compute_gcd(agenda_list.speed[(int)agenda_list.motors],cycle));

current->cycle = cycle;

e_start_timer_processing(agenda_list.motors+1);


return EXIT_OK;
}
else
return AG_NOT_FOUND;
}


/*! \brief Destroy an agenda
 *
 * Destroy the agenda with a given callback function
 * \param func function to test
 * \return int      return the success of the destruction (EXIT_OK for successfull,
 * AG_NOT_FOUND for unsuccessfull).
 */
int e_destroy_agenda(void (*func)(void))
{
Agenda *preceding;
Agenda *current;
int i=0;

for (;i<3;++i) {
preceding = 0;
current = agenda_list.agendas[i];
while (current && current->function != func) {
current = current->next;
preceding = current;
}
if (current) {  // We found a match
if (preceding)
preceding->next = current->next;
else
agenda_list.agendas[i] = current->next;
free(current);
agenda_list.timers_in_use[i]--;
return (EXIT_OK);
}
}
return (AG_NOT_FOUND);
}

/* \brief Change the cycle of an agenda
 *
 * Change the cycle value of an agenda with a given callback function.
 * \param func  function to test
 * \param cycle      new cycle value in millisec/10
 */
int e_set_agenda_cycle(void (*func)(void),unsigned cycle)
{
Agenda *current=agenda_list.waiting;
Agenda *previous=0;
int i, timer;
unsigned gcd;

e_end_agendas_processing(0);


// First check the waiting list to see if func is present if the new cycle is bigger than 0
if (cycle>0) {
while(current && current->function != func) {
previous=current;
current = current->next;
}


if (current) {   // It is in the waiting list. We take it out and assign to one timer
if (previous) {
previous->next=current->next;
}
else {
agenda_list.waiting=current->next;
}
current->next=0;
current->cycle=cycle;
timer=assign_agenda(current);

return EXIT_OK;
}
}

for (i=0;i<3;++i) {
previous=0;
current = agenda_list.agendas[i];
while(current && current->function != func) {
previous=current;
current = current->next;
}
```

```
if (current)
break;
}

if (!current)
return AG_NOT_FOUND;


if (cycle==0) {   // Removal of the task from the list to the waiting list
if (previous)
previous->next=current->next;
else {
  agenda_list.agendas[i]=current->next;
// if (agenda_list.agendas[i]==0)
// e_end_agendas_processing(i+1);
}
agenda_list.timers_in_use[i]--;
current->next=agenda_list.waiting;
agenda_list.waiting=current;
current->cycle=0;
current->activate=0;
}
else {     // Compute the new speed of the timer according

gcd=compute_gcd(cycle,agenda_list.speed[i]);

// e_end_agendas_processing(i+1);
e_set_timer_speed(i+1,gcd);

current->cycle = cycle;

// e_start_timer_processing(i+1);
}
e_start_timer_processing(0);

return EXIT_OK;
}

 /*! \brief Reset an agenda
 *
 * Reset an agenda's counter with a given callback function
 * \param func function to reset
 */
int e_reset_agenda(void (*func)(void))
{
Agenda* current=find_function(func);

if (current) {
current->counter = 0;
return EXIT_OK;
}
else
return AG_NOT_FOUND;

}

/*! \brief Pause an agenda
 *
 * Pause an agenda but do not reset its information.
 * @param func  function to pause
 */
int e_pause_agenda(void (*func)(void))
{
Agenda* current=find_function(func);

if (current) {
current->activate = 0;
return EXIT_OK;
}
else
return AG_NOT_FOUND;
}

/*! \brief Restart an agenda previously paused
 *
 * Restart an agenda previously paused
 * @param func  function to restart
 */
int e_restart_agenda(void (*func)(void))
{
Agenda* current=find_function(func);

if (current) {
current->activate = 1;
return EXIT_OK;
}
else
return AG_NOT_FOUND;
}

/*! \brief Interrupt from timer1
 *
 * Parse the chained list of agenda.
 * \n Increment counter only.
 * \n Check if agenda has to be treated according to the cycle value
 * and current counter value.
 * \n Do it for number of cycle positive or null.
 * \n Check if a service has to be activated.
 */
void __attribute__((__interrupt__, auto_psv))
 _T1Interrupt(void)
{
if (agenda_list.timers_in_use[0] == 0 || agenda_list.speed[0] == 0)
return;

Agenda *current = agenda_list.agendas[0];
```

```
IFS0bits.T1IF = 0;


while (current)
{
// agenda must be active with a positive non-null cycle value
if(current->activate == 1 && current->cycle > 0)
{
current->counter+=agenda_list.speed[0];
// check if the agenda event must be triggered
if(current->counter >= current->cycle) // a cycle value of 1 will be performed every interupt
{
current->function(); // trigger the associated function
current->counter=0; // reset the counter
}
}
current = current->next;
}
current=0;
  return;
}

/*! \brief Interrupt from timer2
 *
 * Parse the chained list of agenda.
 * \n Increment counter only.
 * \n Check if agenda has to be treated according to the cycle value
 * and current counter value.
 * \n Do it for number of cycle positive or null.
 * \n Check if a service has to be activated.
 */
void __attribute__((__interrupt__, auto_psv))
 _T2Interrupt(void)
{

if (agenda_list.timers_in_use[1] == 0  || agenda_list.speed[1] == 0)
return;

Agenda *current = agenda_list.agendas[1];

IFS0bits.T2IF = 0;

while (current)
{
// agenda must be active with a positive non-null cycle value
if(current->activate == 1 && current->cycle > 0)
{
current->counter+=agenda_list.speed[1];
// check if the agenda event must be triggered
if(current->counter > current->cycle-1) // a cycle value of 1 will be performed every interupt
{
current->function(); // trigger the associeted function
current->counter=0; // reset the counter
}
}
current = current->next;
}
  return;
}


/*! \brief Interrupt from timer3
 *
 * Parse the chained list of agenda.
 * \n Increment counter only.
 * \n Check if agenda has to be treated according to the cycle value
 * and current counter value.
 * \n Do it for number of cycle positive or null.
 * \n Check if a service has to be activated.
 */
void __attribute__((__interrupt__, auto_psv))
 _T3Interrupt(void)
{

if (agenda_list.timers_in_use[2] == 0 || agenda_list.speed[2] == 0)
return;

Agenda *current = agenda_list.agendas[2];

IFS0bits.T3IF = 0;

while (current)
{
// agenda must be active with a positive non-null cycle value
if(current->activate == 1 && current->cycle > 0)
{
current->counter+=agenda_list.speed[2];
// check if the agenda event must be triggered
if(current->counter > current->cycle-1) // a cycle value of 1 will be performed every interupt
{
current->function(); // trigger the associeted function
current->counter=0; // reset the counter
}
}
current = current->next;
}
  return;
}

/*void SendAgendaStatus()
{
e_end_agendas_processing(0);
Agenda* current;

unsigned i=0;
for (i=0;i<3;++i) {
sprintf(buffer,"agenda=%u  speed=%u\r\n",i,agenda_list.speed[i]);
```

```
uart_send_text(buffer);
current=agenda_list.agendas[i];
if (!current) {
uart_send_static_text("  Agenda not in use \r\n");
}
else {
while (current) {
sprintf(buffer," %u\r\n",current->cycle);
uart_send_text(buffer);
current=current->next;
}

}
}

current=agenda_list.waiting;
uart_send_static_text("Waiting list : \r\n");

if (!current) {
uart_send_static_text("  Waiting list empty \r\n");
}
else {
while (current) {
sprintf(buffer," %u\r\n",current->cycle);
uart_send_text(buffer);
current=current->next;
}
}


e_start_timer_processing(0);

}

*/
```

2.14.2 e_led.{h,c}

```
/***************************************************************************

fonctions for simple LED manipulation
december 2004: first example for microinfo
by Francesco Mondadaa


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

***************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the LEDs with blinking possibility (timer1, 2, 3).
 *
 * Here we use the fast agenda solution to make the LED blinking.
 * \sa e_agenda.h
 * \author Code: Francesco Mondada \n Doc: Jonathan Besuchet
 */

#ifndef _LED
#define _LED

/* functions */
void e_set_led(unsigned int led_number, unsigned int value); // set led_number (0-7) to value (0-1)
void e_led_clear(void);
void e_blink_led(void);
void e_blink_led0(void);
void e_blink_led1(void);
void e_blink_led2(void);
void e_blink_led3(void);
void e_blink_led4(void);
void e_blink_led5(void);
void e_blink_led6(void);
void e_blink_led7(void);

void e_set_body_led(unsigned int value); // value (0=off 1=on higher=inverse)
void e_set_front_led(unsigned int value); //value (0=off 1=on higher=inverse)

void e_start_led_blinking(int cycle);
void e_stop_led_blinking(void);

#endif
```

```
/***************************************************************************

fonctions for simple LED manipulation
december 2004: first example for microinfo
by Francesco Mondadaa


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada
```

```
Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the LEDs with blinking possibility (timer1, 2, 3).
 *
 * Here we use the fast agenda solution to make the LED blinking.
 * \sa e_agenda.h
 * \author Code: Francesco Mondada \n Doc: Jonathan Besuchet
 */

#include "../../e_epuck_ports.h"
#include "e_agenda_fast.h"


/*! \brief turn on/off the specified LED
 *
 * The e-puck has 8 green LEDs. With this function, you can
 * change the state of these LEDs.
 * \param led_number between 0 and 7
 * \param 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.
 */
void e_set_led(unsigned int led_number, unsigned int value)
// led_number between 0 and 7, value 0 (off) or 1 (on)
// if led_number other than 0-7, all leds are set to value
{
switch(led_number)
{
case 0:
{
if(value>1)
LED0 = LED0^1;
else
LED0 = value;
break;
}
case 1:
{
if(value>1)
LED1 = LED1^1;
else
LED1 = value;
break;
}
case 2:
{
if(value>1)
LED2 = LED2^1;
else
LED2 = value;
break;
}
case 3:
{
if(value>1)
LED3 = LED3^1;
else
LED3 = value;
break;
}
case 4:
{
if(value>1)
LED4 = LED4^1;
else
LED4 = value;
break;
}
case 5:
{
if(value>1)
LED5 = LED5^1;
else
LED5 = value;
break;
}
case 6:
{
if(value>1)
LED6 = LED6^1;
else
LED6 = value;
break;
}
case 7:
{
if(value>1)
LED7 = LED7^1;
else
LED7 = value;
break;
}
default:
LED0 = LED1 = LED2 = LED3 = LED4 = LED5 = LED6 = LED7 = value;
}
}

/*! \brief 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".
 */
void e_led_clear(void)
{
LED0 = 0;
LED1 = 0;
LED2 = 0;
LED3 = 0;
LED4 = 0;
LED5 = 0;
LED6 = 0;
LED7 = 0;
}

/*! \brief 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.
 * \param value 0 (off), 1 (on) otherwise change the state
 */
void e_set_body_led(unsigned int value)
{
if(value>1)
BODY_LED = BODY_LED^1;
else
BODY_LED = value;
}

/*! \brief 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.
 * \param value 0 (off), 1 (on) otherwise change the state
 */
void e_set_front_led(unsigned int value)
{
if(value>1)
FRONT_LED = FRONT_LED^1;
else
FRONT_LED = value;
}

/*! \brief Change the state of all LED
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led(void)
{
LED0 = ~LED0;
LED1 = ~LED1;
LED2 = ~LED2;
LED3 = ~LED3;
LED4 = ~LED4;
LED5 = ~LED5;
LED6 = ~LED6;
LED7 = ~LED7;
}

/*! \brief Change the state of LED0
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led0(void)
{
LED0 = ~LED0;
}

/*! \brief Change the state of LED1
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led1(void)
{
LED1 = ~LED1;
}

/*! \brief Change the state of LED2
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led2(void)
{
LED2 = ~LED2;
}

/*! \brief Change the state of LED3
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led3(void)
{
LED3 = ~LED3;
}

/*! \brief Change the state of LED4
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led4(void)
{
```

```
LED4 = ~LED4;
}

/*! \brief Change the state of LED5
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led5(void)
{
LED5 = ~LED5;
}

/*! \brief Change the state of LED6
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led6(void)
{
LED6 = ~LED6;
}

/*! \brief Change the state of LED7
 *
 * Callback function for an agenda.
 * \sa AgendaType
 */
void e_blink_led7(void)
{
LED7 = ~LED7;
}

/*! \brief Start blinking all LED
 *
 * \param cycle    the number of cycle we wait before launching "e_blink_led()"
 * \sa e_blink_led, e_activate_agenda
 */
void e_start_led_blinking(int cycle)
{
  e_activate_agenda(e_blink_led, cycle);
}

/*! \brief Stop blinking all LED
 *
 * This function use e_destroy_agenda(void (*func)(void))
 * \sa e_destroy_agenda
 */
void e_stop_led_blinking(void)
{
  e_destroy_agenda(e_blink_led);
}

/*! \brief Change the blinking speed
 *
 * This function use e_set_agenda_cycle(void (*func)(void), int cycle)
 * \param cycle    the number of cycle we wait before launching "e_blink_led()"
 * \sa e_blink_led, e_set_agenda_cycle
 */
void e_set_blinking_cycle(int cycle)
{
if (cycle>=0)
e_set_agenda_cycle(e_blink_led,cycle);
}
```

2.14.3 e_motors.{h,c}

```
/*****************************************************************************

Advance control motor of e-puck
December 2004: first version
Basic examples from Lucas Meier & Francesco Mondada
Adaptation, formatting and test by Francesco Mondadaa


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the motors (with timer1, 2, 3)
 *
 * This module manage the motors with the fast agenda solution (timer1, 2, 3).
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#ifndef _MOTORS
#define _MOTORS

/* internal functions */
//void run_left_motor(void);
//void run_right_motor(void);

/* user called function */
void e_init_motors(void);  // init to be done before using the other calls
```

```
void e_set_speed_left(int motor_speed);   // motor speed in percent
void e_set_speed_right(int motor_speed); // motor speed in percent
void e_set_speed(int linear_speed, int angular_speed);

void e_set_steps_left(int steps_left);
void e_set_steps_right(int steps_right);

int e_get_steps_left();
int e_get_steps_right();

#endif
```

```
/*****************************************************************************

Advance control motor of e-puck
December 2004: first version
Basic examples from Lucas Meier & Francesco Mondada
Adaptation, formatting and test by Francesco Mondadaa


This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Francesco Mondada, Lucas Meier

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup motor_LED
 * \brief Manage the motors (with timer1, 2, 3)
 *
 * This module manage the motors with the fast agenda solution (timer1, 2, 3).
 * \sa e_agenda.h
 * \author Code: Francesco Mondada, Lucas Meier \n Doc: Jonathan Besuchet
 */

#include "../../e_epuck_ports.h"
#include "e_agenda_fast.h"
#include <stdlib.h>


/* If powersave is enabled, the motor library will not leave
 * the motor's phase powered when running at low speed, thus reducing the heat
 * dissipation and power consumption
 *
 * TRESHV == the "virtual" speed when powersave is enabled, the phase will
 * be keept on 1/TRESHV seconds.
 *
 * Powersave will only be enabled for speed < MAXV
 */
#define POWERSAVE
#define TRESHV 650
#define MAXV 601

#if TRESHV <= MAXV
#error TRESHV must be higher than MAXV
#endif

/* internal variables */
static int left_speed = 0;
static int right_speed = 0;

static int left_motor_phase=0;     // phase can be 0 to 3
static int right_motor_phase=0;    // phase can be 0 to 3

static int nbr_steps_left=0;
static int nbr_steps_right=0;

/*------ internal calls ------*/

/*! Change left motor phase according to the left_speed signe. */
 void run_left_motor(void)  // interrupt for motor 1 (of two) = left motor
{
  // increment or decrement phase depending on direction

#ifdef POWERSAVE
  static int phase_on = 0;
  if(phase_on && abs(left_speed) < MAXV) {
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
MOTOR1_PHC = 0;
MOTOR1_PHD = 0;
phase_on = 0;
e_set_motor_speed(run_left_motor, 10000/abs(left_speed) - 10000/TRESHV);
return;
  }
#endif

  if (left_speed > 0) // inverted for the two motors
  {
    nbr_steps_left++;
    left_motor_phase--;
    if (left_motor_phase < 0) left_motor_phase = 3;
  }
  else
  {
    nbr_steps_left--;
```

```
    left_motor_phase++;
    if (left_motor_phase > 3) left_motor_phase = 0;
  }

  // set the phase on the port pins

  switch (left_motor_phase)
  {
    case 0:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR1_PHA = 0;
      MOTOR1_PHB = 1;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 1;
      MOTOR1_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR1_PHA = 1;
      MOTOR1_PHB = 0;
      MOTOR1_PHC = 0;
      MOTOR1_PHD = 1;
      break;
    }
  }
#ifdef POWERSAVE
  if(abs(left_speed) < MAXV) {
    phase_on = 1;
    e_set_motor_speed(run_left_motor,10000/TRESHV);
  }
#endif

}

/*! Change right motor phase according to the right_speed signe */
void run_right_motor(void)  // interrupt for motor 2 (of two) = right motor
{
  // increment or decrement phase depending on direction

#ifdef POWERSAVE
  static int phase_on = 0;
  if(phase_on && abs(right_speed) < MAXV) {
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
MOTOR2_PHC = 0;
MOTOR2_PHD = 0;
phase_on = 0;
e_set_motor_speed(run_right_motor, 10000/abs(right_speed) - 10000/TRESHV);
return;
  }
#endif


  if (right_speed < 0)
  {
  nbr_steps_right++;
    right_motor_phase--;
    if (right_motor_phase < 0) right_motor_phase = 3;
  }
  else
  {
    nbr_steps_right--;
    right_motor_phase++;
    if (right_motor_phase > 3) right_motor_phase = 0;
  }

  // set the phase on the port pins

  switch (right_motor_phase)
  {
    case 0:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
    case 1:
    {
      MOTOR2_PHA = 0;
      MOTOR2_PHB = 1;
      MOTOR2_PHC = 1;
      MOTOR2_PHD = 0;
      break;
    }
    case 2:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 1;
```

```
      MOTOR2_PHD = 0;
      break;
    }
    case 3:
    {
      MOTOR2_PHA = 1;
      MOTOR2_PHB = 0;
      MOTOR2_PHC = 0;
      MOTOR2_PHD = 1;
      break;
    }
  }
#ifdef POWERSAVE
  if(abs(right_speed) < MAXV) {
    phase_on = 1;
    e_set_agenda_cycle(run_right_motor,10000/TRESHV);
  }
#endif

}

/* ---- user calls ---- */

/*! \brief 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.
 * \sa e_activate_agenda
 */
void e_init_motors(void)
{
#ifndef __AGENDA_FAST_H__
  e_activate_agenda(run_left_motor, 0);
  e_activate_agenda(run_right_motor, 0);
#else
e_activate_motors(run_left_motor,run_right_motor);
#endif
}

/*! \brief 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)").
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 * \sa e_set_agenda_cycle
 */
void e_set_speed_left(int motor_speed)  // motor speed in percent
{
  // speed null
  if (motor_speed == 0)
  {
    left_speed = 0;

#ifdef __AGENDA_FAST_H__
e_set_motor_speed(run_left_motor, 0);
#else
    e_set_agenda_cycle(run_left_motor, 0);
#endif
    MOTOR1_PHA = 0;
    MOTOR1_PHB = 0;
    MOTOR1_PHC = 0;
    MOTOR1_PHD = 0;
  }
  // speed inferior to the minimum value
  else if(motor_speed < -1000)
  {
    left_speed = -1000;
#ifdef __AGENDA_FAST_H__
e_set_motor_speed(run_left_motor, (int)-10000/left_speed);
#else
    e_set_agenda_cycle(run_left_motor, (int)-10000/left_speed);
#endif
  }
  // speed superior to the maximum value
  else if(motor_speed > 1000)
  {
    left_speed = 1000;
#ifdef __AGENDA_FAST_H__
e_set_motor_speed(run_left_motor, (int)10000/left_speed);
#else
    e_set_agenda_cycle(run_left_motor, (int) 10000/left_speed);
#endif
  }
  else
  {
    left_speed = motor_speed;
  // negative speed
if(motor_speed < 0) {
#ifdef __AGENDA_FAST_H__
e_set_motor_speed(run_left_motor, (int)-10000/motor_speed);
#else
e_set_agenda_cycle(run_left_motor, (int)-10000/motor_speed);
#endif
}
  // positive speed
else {
#ifdef __AGENDA_FAST_H__
e_set_motor_speed(run_left_motor, (int) 10000/motor_speed);
#else
e_set_agenda_cycle(run_left_motor, (int) 10000/motor_speed);
#endif
}
  }
}
```

```
/*! \brief 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)").
 * \param motor_speed from -1000 to 1000 give the motor speed in steps/s,
 * positive value to go forward and negative to go backward.
 * \sa e_set_agenda_cycle
 */
void e_set_speed_right(int motor_speed)  // motor speed in percent
{
  // speed null
  if (motor_speed == 0)
  {
    right_speed = 0;
#ifdef __AGENDA_FAST_H__
e_set_motor_speed(run_right_motor, 0);
#else
    e_set_agenda_cycle(run_right_motor, 0);
#endif
    MOTOR2_PHA = 0;
    MOTOR2_PHB = 0;
    MOTOR2_PHC = 0;
    MOTOR2_PHD = 0;
  }
  // speed inferior to the minimum value
  else if (motor_speed < -1000)
  {
    right_speed = -1000;
#ifdef __AGENDA_FAST_H_
e_set_motor_speed(run_right_motor, (int)-10000/right_speed);
#else
    e_set_agenda_cycle(run_right_motor, (int)-10000/right_speed);
#endif
  }
  // speed superior to the maximum value
  else if (motor_speed > 1000)
  {
    right_speed = 1000;
#ifdef __AGENDA_FAST_H_
e_set_motor_speed(run_right_motor, (int) 10000/right_speed);
#else
    e_set_agenda_cycle(run_right_motor, (int) 10000/right_speed);
#endif
  }
  else
  {
    right_speed = motor_speed;

  // negative speed
if(motor_speed < 0){
#ifdef __AGENDA_FAST_H_
e_set_motor_speed(run_right_motor, (int)-10000/motor_speed);
#else
e_set_agenda_cycle(run_right_motor, (int)-10000/motor_speed);
#endif
}
  // positive speed
else {
#ifdef __AGENDA_FAST_H_
e_set_motor_speed(run_right_motor, (int) 10000/motor_speed);
#else
e_set_agenda_cycle(run_right_motor, (int) 10000/motor_speed);
#endif
}

  }
}

/*! \brief Manage linear/angular speed
 *
 * This function manage the speed of the motors according to the
 * desired linear and angular speed.
 * \param linear_speed the speed in the axis of e-puck
 * \param angular_speed the rotation speed (trigonometric)
 */
void e_set_speed(int linear_speed, int angular_speed)
{
if(abs(linear_speed) + abs(angular_speed) > 1000)
return;
else
{
e_set_speed_left (linear_speed - angular_speed);
e_set_speed_right(linear_speed + angular_speed);
}
}

/*! \brief Give the number of left motor steps
 * \return The number of phases steps made since the left motor
 * is running.
 */
int e_get_steps_left()
{
return nbr_steps_left;
}

/*! \brief Set the number of left motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_left(int set_steps)
{
nbr_steps_left=steps_left;
}

/*! \brief Give the number of right motor steps
 * \return The number of phases steps made since the right motor
 * is running.
```

```
 */
int e_get_steps_right()
{
return nbr_steps_right;
}

/*! \brief Set the number of right motor steps
 * \param set_steps The number of changed phases that you want set.
 */
void e_set_steps_right(int set_steps)
{
nbr_steps_right=steps_right;
}
```

2.15 uart

2.15.1 e_epuck_ports.inc

```
;*************************************************************
;  Defintition of all port of e-puck   *
; Version 1.0 november 2005 *
; Michael Bonani,  *
; *
;*************************************************************

;.IFNDEF _EPUCK_PORTS
;.def  _EPUCK_PORTS

.include "p30f6014A.inc"

;/*********************GENERAL SETUP************************/

.equiv FOSC, 7372800     ; 7.3728Mhz crystal in XTL mode
.equiv PLL, 8        ; 8x PLL


.equiv FCY,     ((FOSC*PLL)/(4)) ; Instruction cycle frequency
.equiv MILLISEC , (FCY/1000) ; 1mSec delay constant
.equiv MICROSEC,  (FCY/1000000) ; 1uSec delay constant
.equiv NANOSEC,   (FCY/1000000000) ; 1nSec delay constant

.equiv TCY_PIC,(1000000000/FCY) ;time instruction cycle in [ns]
.equiv INTERRUPT_DELAY,(10*TCY_PIC);delay to start an interrupt in [ns] (observe with p30f6014)

.equiv TRUE,1
.equiv FALSE,0


;/********************** OUTPUTS ***************************/
.equiv OUTPUT_PIN, 0
;/*LEDS*/
;/*First in front of robot than turning clokwise*/
.equiv LEDPORT, LATA
.equiv LED0, LATA6
.equiv LED1, LATA7
.equiv LED2, LATA9
.equiv LED3, LATA12
.equiv LED4, LATA10
.equiv LED5, LATA13
.equiv LED6, LATA14
.equiv LED7, LATA15

.macro ONLED num
bset LEDPORT,#\num
.endm
.macro OFFLED num
bclr LEDPORT,#\num
.endm

.equiv LEDPORT2, LATC
.equiv FRONT_LED, LATC1
.equiv BODY_LED, LATC2


;/*IR*/
.equiv PULSE_PORT1, LATF
.equiv PULSE_IR0, LATF7 ; PULSE IR 0 and 4
.equiv PULSE_IR1, LATF8 ; PULSE IR 1 and 5
.equiv PULSE_PORT2, LATg
.equiv PULSE_IR2, LATG0 ; PULSE IR 2 and 6
.equiv PULSE_IR3, LATG1 ; PULSE IR 3 and 7


;/*First in front left of robot than turning clokwise*/
.equiv IR0, 8  ; ir proximity sensor 0 on AD channel 8
.equiv IR1, 9  ; ir proximity sensor 1 on AD channel 9
.equiv IR2, 10  ; ir proximity sensor 2 on AD channel 10
.equiv IR3, 11  ; ir proximity sensor 3 on AD channel 11
.equiv IR4, 12  ; ir proximity sensor 4 on AD channel 12
.equiv IR5, 13  ; ir proximity sensor 5 on AD channel 13
.equiv IR6, 14  ; ir proximity sensor 6 on AD channel 14
.equiv IR7, 15  ; ir proximity sensor 7 on AD channel 15


;/*analog*/
.equiv MIC1, 2  ; microphone 1 on AD channel 2
.equiv MIC2, 3  ; microphone 2 on AD channel 3
.equiv MIC3, 4  ; microphone 3 on AD channel 4

.equiv ACCX, 5  ; X Axis of accelerometer on AD channel 5
.equiv ACCY, 6  ; Y Axis of accelerometer on AD channel 6
.equiv ACCZ, 7  ; Z Axis of accelerometer on AD channel 7
```

```
;/*basic audio*/
.equiv audio_on, LATF0


;/*motors*/
.equiv MOTOR_PORT, LATD
.equiv MOTOR1_PHA, LATD0
.equiv MOTOR1_PHB, LATD1
.equiv MOTOR1_PHC, LATD2
.equiv MOTOR1_PHD, LATD3
.equiv MOTOR2_PHA, LATD4
.equiv MOTOR2_PHB, LATD5
.equiv MOTOR2_PHC, LATD6
.equiv MOTOR2_PHD, LATD7



;/*camera*/
.equiv CAM_RESET, LATC13

;/* I2C */
.equiv SIO_D,LATG3
.equiv SIO_C,LATG2


;/********************** INPUTS *************************/
.equiv INPUT_PIN, 1

;/*low battery signal active low when Vbatt<3.4V*/
.equiv BATT_LOW, RF1


;/* selector on normal extension*/
.equiv SELECTOR_PORT, PORTG
.equiv SELECTOR0, RG6
.equiv SELECTOR1, RG7
.equiv SELECTOR2, RG8
.equiv SELECTOR3, RG9


;/*IR TV receiver on normal extension*/
.equiv REMOTE, RF6


;/*CAMERA*/
;/*data higher 8 bits of port D*/
.equiv CAM_DATA, PORTD;

.equiv CAM_Y0, RD8
.equiv CAM_Y1, RD9
.equiv CAM_Y2, RD10
.equiv CAM_Y3, RD11
.equiv CAM_Y4, RD12
.equiv CAM_Y5, RD13
.equiv CAM_Y6, RD14
.equiv CAM_Y7, RD15


;/*clock interupt*/
.equiv CAM_PWDN, RC2
.equiv CAM_VSYNC, RC4
.equiv CAM_HREF, RC3
.equiv CAM_PCLK, RC14


; .ENDIF
```

2.15.2 e_init_uart1.s

```
/*****************************************************************************************************************

Title: UART1_RX_CHAR.s

Author: Inspired Microchip library
Bonani Michael
Stephane Magnenat

History:
21/12/05 Start day
12/03/07 added rx buffer

*****************************************************************************************************************/
;to be used with uart_txrx_char.h

.include "e_epuck_ports.inc"


.equiv BAUDRATE, 115000

.extern _U1RXRcvCnt
.extern _U1RXReadCnt

.section .text

.global _e_init_uart1
_e_init_uart1:
; Init uart 1 at 11500bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX
; Initialize and Enable UART1 for Tx and Rx
clr     U1MODE                          ; Set SFRs to a known state
clr     U1STA
bclr    U1STA, #URXISEL1
bclr    U1STA, #URXISEL0

bclr    IFS0, #U1RXIF
bset    IEC0, #U1RXIE                   ; Enable Rx ISR processing
```

```
bset    U1MODE, #UARTEN                     ; Enable UART
mov     #(((FCY/BAUDRATE)/16)-1), W0        ; Initialize Baud rate
mov     w0, U1BRG                           ; to 115.2 Kbaud

bclr    IFS0, #U1TXIF                       ; Enable Txmit ISR processing
bset    IEC0, #U1TXIE

; set a higher priority on UART interrupt ( so an ISR can use uart )
; uart is slow, don't abuse it in interrupt
bclr.b INTCON1+1,#7
mov IPC2, w0
mov #0xF00F, w1
and w1,w0,w0
mov #0x0550, w1
add w1,w0,w0
mov w0,IPC2

bset    U1STA, #UTXISEL
bset    U1STA, #UTXEN                       ; Enable Transmission

; Reception counters to 0
clr _U1RXRcvCnt
clr _U1RXReadCnt
return


.end
```

2.15.3 e_init_uart2.s

```
/*****************************************************************************************************************

Title: UART1_RX_CHAR.s

Author: Inspired Microchip library
Bonani Michael

History:
21/12/05 Start day
12/03/07 added rx buffer

*****************************************************************************************************************/
;to be used with uart_txrx_char.h

.include "e_epuck_ports.inc"
.equiv BAUDRATE, 115000

.extern _U2RXRcvCnt
.extern _U2RXReadCnt

.section .text

.global _e_init_uart2
_e_init_uart2:
; Init uart 2 at 11500bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX
; Initialize and Enable UART1 for Tx and Rx
clr     U2MODE                              ; Set SFRs to a known state
clr     U2STA
bclr    U2STA, #URXISEL1
bclr    U2STA, #URXISEL0

bclr    IFS1, #U2RXIF
bset    IEC1, #U2RXIE                       ; Enable Rx ISR processing

bset    U2MODE, #UARTEN                     ; Enable UART
mov     #(((FCY/BAUDRATE)/16)-1), W0        ; Initialize Baud rate
mov     w0, U2BRG                           ; 9600 to 115200 Kbaud

bclr    IFS1, #U2TXIF                       ; Enable Txmit ISR processing
bset    IEC1, #U2TXIE

; set a higher priority on UART interrupt ( so an ISR can use uart )
; uart is slow, don't abuse it in interrupt
bclr.b INTCON1+1,#7
mov IPC6, w0
mov #0xFF00, w1
and w1,w0,w0
mov #0x0055, w1
add w1,w0,w0
mov w0,IPC2

bset    U2STA, #UTXISEL
bset    U2STA, #UTXEN                       ; Enable Transmission

; Reception counters to 0
clr _U2RXRcvCnt
clr _U2RXReadCnt
return


.end
```

2.15.4 e_uart_char.h

```
/*****************************************************************************

UART module
December 2004: first version Michael Bonani
```

```
This file is part of the e-puck library license.
See http://www.e-puck.org/index.php?option=com_content&task=view&id=18&Itemid=45

(c) 2004-2007 Michael Bonani

Robotics system laboratory http://lsro.epfl.ch
Laboratory of intelligent systems http://lis.epfl.ch
Swarm intelligent systems group http://swis.epfl.ch
EPFL Ecole polytechnique federale de Lausanne http://www.epfl.ch

*****************************************************************************/

/*! \file
 * \ingroup uart
 * \brief Manage UART.
 *
 * This module manage all the UART ressource.
 * \n 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.
 * \code
 * #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);
 *  }
 * }
 * \endcode
 * \author Code: Michael Bonani \n Doc: Jonathan Besuchet
 */

/*! \defgroup uart UART
 *
 * \section intro_sec 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).
 * \n Even these files are written in ASM, you can call them by including the e_uart_char.h
 * files in your C code (look at the exemple in e_uart_char.h).
 * \section bluettothUart_sec 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: \ref bluetooth
 * - 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
 */

#ifndef _UART_TXRX_CHAR
#define _UART_TXRX_CHAR

/*! \brief Init uart 1 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX */
void e_init_uart1(void);

/*! \brief Check if something is comming on uart 1
 * \return the number of characters available, 0 if none are available */
int  e_ischar_uart1();

/*! \brief If available, read 1 char and put it in pointer
 * \param car The pointer where the caracter will be stored if available
 * \return 1 if a char has been readed, 0 if no char is available
 */
int  e_getchar_uart1(char *car);

/*! \brief Send a buffer of char of size length
 * \param buff The top of the array where the data are stored
 * \param length The length of the array to send
 */
void e_send_uart1_char(const char * buff, int length);

/*! \brief  To check if the sending operation is done
 * \return 1 if buffer sending is in progress, return 0 if not
 */
int  e_uart1_sending(void);
```

```
/*! \brief Init uart 2 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX */
void e_init_uart2(void);

/*! \brief Check if something is comming on uart 2
 * \return the number of characters available, 0 if none are available */
int  e_ischar_uart2();

/*! \brief If available, read 1 char and put it in pointer
 * \param car The pointer where the caracter will be stored if available
 * \return 1 if a char has been readed, 0 if no char is available
 */
int  e_getchar_uart2(char *car);

/*! \brief Send a buffer of char of size length
 * \param buff The top of the array where the datas are stored
 * \param length The length of the array
 */
void e_send_uart2_char(const char * buff, int length);

/*! \brief  To check if the sending operation is done
 * \return 1 if buffer sending is in progress, return 0 if not
 */
int  e_uart2_sending(void);


extern void *e_uart1_int_clr_addr; //address to be clear on interrupt
extern int e_uart1_int_clr_mask; //mask to be use to clear on interrupt
extern void *e_uart2_int_clr_addr; //address to be clear on interrupt
extern int e_uart2_int_clr_mask; //mask to be use to clear on interrupt


#endif
```

2.15.5 e_uart1_rx_char.s

```
/************************************************************************************************************************

Title: UART1_RX_CHAR.s

Author: Inspired Microchip library
Davis Daidi
Bonani Michael
Stephane Magnenat

History:
11/07/05 Start day
18/07/05 To be continued
21/12/05 Optimisation and new function name
12/02/07 Added possibility of clearing an external on interrupt
12/03/07 Major rewrite, added rx buffer

************************************************************************************************************************/

; to be used with e_uart_char.h

.include "p30f6014A.inc"

.section  .data, near

.global _U1RXBuf
.global _U1RXRcvCnt
.global _U1RXReadCnt
_U1RXBuf: .space 64 ; reception buffer
_U1RXRcvCnt: .space 2 ; amount of received bytes
_U1RXReadCnt: .space 2 ; amount of read bytes

.ifdef UART1_CLR_BIT_ON_INT
.align 2
.global _e_uart1_int_clr_mask
.global _e_uart1_int_clr_addr
_e_uart1_int_clr_mask: .space 2 ; mask to apply upon interrupt
_e_uart1_int_clr_addr: .space 2 ; pointer where to apply mask upon interrupt
.endif

.section  .text


.global __U1RXInterrupt
__U1RXInterrupt:
push.d w0 ; Save context - w0, w1

bclr    IFS0, #U1RXIF ; Clear the interrupt flag.

.ifdef UART1_CLR_BIT_ON_INT
mov _e_uart1_int_clr_addr, w1 ; Load pointer in w0
mov [w1], w0 ; Deference pointer read
and _e_uart1_int_clr_mask ; Mask w0 with user-specified mask
mov w0, [w1] ; Deference pointer write
.endif

mov _U1RXRcvCnt, w0 ; Received counter in w0
and #0x3f, w0 ; Mask at buffer length
mov #_U1RXBuf, w1 ; Buffer pointer in w1
add w0, w1, w1 ; Element pointer in w1
clr     w0 ; Clear w0
mov.b   U1RXREG, WREG ; Transfer received byte to w0
mov.b w0, [w1] ; Store received byte
inc _U1RXRcvCnt ; Increment amount of received bytes

pop.d   w0 ; Restore context - w0, w1

retfie ; Return from Interrupt


.global _e_ischar_uart1
```

```
_e_ischar_uart1:
mov _U1RXRcvCnt, w0 ; Received counter in w0
mov _U1RXReadCnt, w1 ; Read counter in w1
sub w0, w1, w0 ; Diff (amount of unread) in w0
return


.global _e_getchar_uart1
_e_getchar_uart1:
mov _U1RXRcvCnt, w1 ; Received counter in w1
mov _U1RXReadCnt, w2 ; Read counter in w2
cp w1, w2 ; Compare w1 - w2

bra Z, no_char_to_ret ; If equal, no char to return

and #0x3f, w2 ; Mask at buffer length
mov #_U1RXBuf, w1 ; Buffer pointer in w1
add w1, w2, w2 ; Element pointer in w2

mov.b [w2], w1 ; Read byte from reception buffer
mov.b w1, [w0] ; Store byte to user buffer

inc _U1RXReadCnt ; Increment amount of read bytes

mov #1, w0 ; Return 1
return

no_char_to_ret:
clr w0 ; Return 0
return

.end ; EOF
```

2.15.6 e_uart1_tx_char.s

```
/************************************************************************************************************************

Title: uart1_tx_char.s

Author: Inspired Microchip library
Davis Daidi
Michael Bonani
History:
11/07/05 Start day
16/09/05 Adaptation uart2->uart1
22/12/05 Optimisation and chand name of function

************************************************************************************************************************/
;to be used with uart_txrx_char.h

.include "p30f6014A.inc"


.global U1TXPtr
.global U1TXLength
.global U1TXOk

.section  .data, near

U1TXPtr:  .hword 0x0000
U1TXLength: .hword 0x0000
U1TXOk: .hword 0x0000

.global         __U1TXInterrupt

.section        .text
__U1TXInterrupt:
bclr    IFS0, #U1TXIF           ;Clear the interrupt flag
        push.d  w0                      ;save context - w0,w1, w2
        push    w2


cp0 U1TXOk
bra z, end_transmission

        cp0     U1TXPtr                 ;Check if the pointer is valid (not zero)
        bra     z, exit2_U1TXInt        ;Else exit

cp0 U1TXLength
bra z,exit2_U1TXInt

        mov     U1TXPtr, w1
        mov     #4, w2                  ;Initialize w2=4 as a decrementing counter

; make sure the buffer is empty

load_next_char:

btsc U1STA,#9 ; make sure the buffer is not full
bra  load_next_char

        mov U1TXLength, w0 ;load lenght of buffer
cp0     w0                      ;unless end of buffer is encountered
        bra     z, loaded_all_char
dec w0,w0
mov  WREG,U1TXLength ;rest of lenght

mov.b   [w1++], w0              ;Load 4 characters into UART TX queue
        mov.b   WREG, U1TXREG
        dec     w2, w2
        bra     nz, load_next_char
```

```
        bra     exit1_U1TXInt           ;Exit ISR after loading 4 chars

loaded_all_char:
        clr     U1TXPtr                 ;Clear pointer if all char send
        bra     end_transmission

exit1_U1TXInt:
        mov     w1, U1TXPtr             ;Save new value of pointer
end_transmission:
  pop     w2                      ;Restore context - w0, w1, w2
        pop.d   w0
        retfie                          ;Return from Interrupt

exit2_U1TXInt:
        mov #0,w0
mov w0,U1TXOk
pop     w2                      ;Restore context - w0, w1, w2
        pop.d   w0
        retfie                          ;Return from Interrupt


.global _e_send_uart1_char

; in: w0 pointer char hon buffer
; in: w1 lengh of buffer

_e_send_uart1_char:
wait_1: cp0 U1TXOk ; check if pointer is under trsnfer
bra     nz, wait_1
bclr    IEC0, #U1TXIE  ;disable interupt
dec w1,w1
mov w1, U1TXLength   ; move lengh of buffer in U2TXLength
mov.b [w0++],w1
mov  w1,U1TXREG
mov     w0, U1TXPtr
mov #1,w1
mov w1,U1TXOk
bset    IEC0, #U1TXIE ;enable interupt
        return

.global _e_uart1_sending

; in: w0 pointer char hon buffer
; in: w1 lengh of buffer

_e_uart1_sending:
mov U1TXOk,w0
        return


.end                                    ;EOF
```

2.15.7 e_uart2_rx_char.s

```
/************************************************************************************************************************

Title: UART2_RX_CHAR.s

Author: Inspired Microchip library
Davis Daidi
Bonani Michael
Stephane Magnenat

History:
11/07/05 Start day
18/07/05 To be continued
21/12/05 Optimisation and new function name
12/02/07 Added possibility of clearing an external on interrupt

************************************************************************************************************************/

; to be used with uart_txrx_char.h

.include "p30f6014a.inc"

.section  .data, near

.global _U2RXBuf
.global _U2RXRcvCnt
.global _U2RXReadCnt
_U2RXBuf: .space 64 ; reception buffer
_U2RXRcvCnt: .space 2 ; amount of received bytes
_U2RXReadCnt: .space 2 ; amount of read bytes

.ifdef UART2_CLR_BIT_ON_INT
.align 2
.global _e_uart2_int_clr_mask
.global _e_uart2_int_clr_addr
_e_uart2_int_clr_mask: .space 2 ; mask to apply upon interrupt
_e_uart2_int_clr_addr: .space 2 ; pointer where to apply mask upon interrupt
.endif

.section  .text


.global __U2RXInterrupt
__U2RXInterrupt:
push.d w0 ; Save context - w0, w1

bclr    IFS0, #U2RXIF ; Clear the interrupt flag.
```

```
.ifdef UART2_CLR_BIT_ON_INT
mov _e_uart2_int_clr_addr, w1 ; Load pointer in w0
mov [w1], w0 ; Deference pointer read
and _e_uart2_int_clr_mask ; Mask w0 with user-specified mask
mov w0, [w1] ; Deference pointer write
.endif

mov _U2RXRcvCnt, w0 ; Received counter in w0
and #0x3f, w0 ; Mask at buffer length
mov #_U2RXBuf, w1 ; Buffer pointer in w1
add w0, w1, w1 ; Element pointer in w1
clr     w0 ; Clear w0
mov.b   U2RXREG, WREG ; Transfer received byte to w0
mov.b w0, [w1] ; Store received byte
inc _U2RXRcvCnt ; Increment amount of received bytes

pop.d   w0 ; Restore context - w0, w1

retfie ; Return from Interrupt


.global _e_ischar_uart2
_e_ischar_uart2:
mov _U2RXRcvCnt, w0 ; Received counter in w0
mov _U2RXReadCnt, w1 ; Read counter in w1
sub w0, w1, w0 ; Diff (amount of unread) in w0
return


.global _e_getchar_uart2
_e_getchar_uart2:
mov _U2RXRcvCnt, w1 ; Received counter in w1
mov _U2RXReadCnt, w2 ; Read counter in w2
cp w1, w2 ; Compare w1 - w2

bra Z, no_char_to_ret ; If equal, no char to return

and #0x3f, w2 ; Mask at buffer length
mov #_U2RXBuf, w1 ; Buffer pointer in w1
add w1, w2, w2 ; Element pointer in w2

mov.b [w2], w1 ; Read byte from reception buffer
mov.b w1, [w0] ; Store byte to user buffer

inc _U2RXReadCnt ; Increment amount of read bytes

mov #1, w0 ; Return 1
return

no_char_to_ret:
clr w0 ; Return 0
return

.end ; EOF
```

2.15.8 e_uart2_tx_char.s

```
/************************************************************************************************************

Title: uart2_tx_char.s

Author: Inspired Microchip library
Davis Daidi
Michael Bonani
History:
11/07/05 Start day
16/09/05 Adaptation uart2->uart1
22/12/05 Optimisation and change name of function

************************************************************************************************************/
;to be used with uart_txrx_char.h

.include "p30f6014a.inc"


.global U2TXPtr
.global U2TXLength
.global U2TXOk

.section  .data, near

U2TXPtr:  .hword 0x0000
U2TXLength: .hword 0x0000
U2TXOk: .hword 0x0000

.global         __U2TXInterrupt

.section        .text
__U2TXInterrupt:
bclr    IFS1, #U2TXIF           ;Clear the interrupt flag
        push.d  w0                      ;save context - w0,w1, w2
        push    w2


cp0 U2TXOk
bra z, end_transmission

        cp0     U2TXPtr                 ;Check if the pointer is valid (not zero)
        bra     z, exit2_U2TXInt        ;Else exit

cp0 U2TXLength
bra z,exit2_U2TXInt
```

```
        mov     U2TXPtr, w1
        mov     #4, w2                  ;Initialize w2=4 as a decrementing counter


load_next_char:

btsc U2STA,#9 ;make sur buffer is not full
bra  load_next_char

        mov U2TXLength, w0 ;load lenght of buffer
cp0     w0                      ;unless end of buffer is encountered
        bra     z, loaded_all_char
dec w0,w0
mov  WREG,U2TXLength ;rest of lenght

mov.b   [w1++], w0              ;Load 4 characters into UART TX queue
        mov.b   WREG, U2TXREG
        dec     w2, w2
        bra     nz, load_next_char
        bra     exit1_U2TXInt           ;Exit ISR after loading 4 chars

loaded_all_char:
        clr     U2TXPtr                 ;Clear pointer if all char send
        bra     end_transmission

exit1_U2TXInt:
        mov     w1, U2TXPtr             ;Save new value of pointer
end_transmission:
  pop     w2                    ;Restore context - w0, w1, w2
        pop.d   w0
        retfie                          ;Return from Interrupt

exit2_U2TXInt:
        mov #0,w0
mov w0,U2TXOk
pop     w2                      ;Restore context - w0, w1, w2
        pop.d   w0
        retfie                          ;Return from Interrupt


.global _e_send_uart2_char

; in: w0 pointer on char buffer
; in: w1 lengh of buffert

_e_send_uart2_char:


wait_1: ; wait until old buffer sent
cp0 U2TXOk ; check if pointer is under trsnfer
bra     nz, wait_1
bclr    IEC1, #U2TXIE  ;disable interupt
dec w1,w1
mov w1, U2TXLength   ; move lengh of buffer in U2TXLength

mov.b [w0++],w1
mov  w1,U2TXREG
mov     w0, U2TXPtr
mov #1,w1
mov w1,U2TXOk
bset    IEC1, #U2TXIE ;enable interupt
        return

.global _e_uart2_sending

_e_uart2_sending:
mov U2TXOk,w0
        return

.end                                    ;EOF
```