Contents

1 e-puck standard library documentation .................................................. 1
   1.1 Introduction ......................................................................................... 1
   1.2 Documentation organization ............................................................... 1
   1.3 External links ....................................................................................... 2

2 e-puck Module Index .................................................................................. 3
   2.1 e-puck Modules .................................................................................... 3

3 e-puck Directory Hierarchy ........................................................................ 5
   3.1 e-puck Directories ................................................................................. 5

4 e-puck Data Structure Index ...................................................................... 7
   4.1 e-puck Data Structures ....................................................................... 7

5 e-puck File Index ......................................................................................... 9
   5.1 e-puck File List .................................................................................... 9

6 e-puck Module Documentation .................................................................. 11
   6.1 Analogic/Digital conversion (ADC) ....................................................... 11
   6.2 Bluetooth ............................................................................................ 14
   6.3 Camera fast two timers ....................................................................... 15
   6.4 Camera slow three timers ................................................................... 18
   6.5 Sound .................................................................................................. 21
   6.6 LIS sensor turret .................................................................................. 23
   6.7 Radio communication .......................................................................... 24
   6.8 FFT ...................................................................................................... 25
   6.9 I2C ....................................................................................................... 27
   6.10 Matlab communication ....................................................................... 31
   6.11 Ports, motors and LEDs ..................................................................... 32
   6.12 UART ................................................................................................. 36
# CONTENTS

## e-puck Directory Documentation

7.1 a_d/ Directory Reference .................................................. 37
7.2 a_d/advance_ad_scan/ Directory Reference .............................. 38
7.3 motor_led/advance_one_timer/ Directory Reference .................... 39
7.4 bluetooth/ Directory Reference ........................................... 40
7.5 camera/ Directory Reference .............................................. 41
7.6 codec/ Directory Reference ................................................ 42
7.7 contrib/ Directory Reference ............................................. 43
7.8 camera/fast_2_timer/ Directory Reference ............................... 44
7.9 motor_led/advance_one_timer/fast_agenda/ Directory Reference .... 45
7.10 fft/ Directory Reference .................................................. 46
7.11 I2C/ Directory Reference ................................................ 47
7.12 contrib/LIS_sensors_turret/ Directory Reference .................... 48
7.13 matlab/ Directory Reference ............................................. 49
7.14 matlab/matlab files/ Directory Reference ............................... 50
7.15 motor_led/ Directory Reference ........................................ 51
7.16 camera/slow_3_timer/ Directory Reference ............................. 52
7.17 contrib/SWIS_com_module/ Directory Reference ....................... 53
7.18 uart/ Directory Reference ............................................... 54

## e-puck Data Structure Documentation

8.1 AgendaList Struct Reference ............................................. 55
8.2 AgendaType Struct Reference .......................................... 57
8.3 BtDevice Struct Reference .............................................. 59
8.4 BtEPuck Struct Reference .............................................. 60
8.5 TypeAccRaw Struct Reference ........................................... 61
8.6 TypeAccSpheric Struct Reference ....................................... 62

## e-puck File Documentation

9.1 a_d/advance_ad_scan/e_acc.c File Reference .......................... 63
9.2 a_d/advance_ad_scan/e_acc.h File Reference .......................... 69
9.3 a_d/advance_ad_scan/e_ad_conv.c File Reference ..................... 74
9.4 a_d/e_ad_conv.c File Reference ......................................... 77
9.5 a_d/advance_ad_scan/e_ad_conv.h File Reference ..................... 78
9.6 a_d/e_ad_conv.h File Reference .......................................... 81
9.7 a_d/advance_ad_scan/e_micro.c File Reference ....................... 82
9.8 a_d/e_micro.c File Reference ........................................... 84
<table>
<thead>
<tr>
<th>Section</th>
<th>File Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.9</td>
<td>a_d/advance_ad_scan/e_micro.h</td>
<td>86</td>
</tr>
<tr>
<td>9.10</td>
<td>a_d/e_micro.h</td>
<td>88</td>
</tr>
<tr>
<td>9.11</td>
<td>a_d/advance_ad_scan/e_prox.c</td>
<td>90</td>
</tr>
<tr>
<td>9.12</td>
<td>a_d/e_prox.c</td>
<td>93</td>
</tr>
<tr>
<td>9.13</td>
<td>a_d/advance_ad_scan/e_prox.h</td>
<td>96</td>
</tr>
<tr>
<td>9.14</td>
<td>a_d/e_prox.h</td>
<td>99</td>
</tr>
<tr>
<td>9.15</td>
<td>a_d/e_accelerometer.c</td>
<td>102</td>
</tr>
<tr>
<td>9.16</td>
<td>a_d/e_accelerometer.h</td>
<td>104</td>
</tr>
<tr>
<td>9.17</td>
<td>a_d/e_prox_timer2.c</td>
<td>106</td>
</tr>
<tr>
<td>9.18</td>
<td>bluetooth/e_bluetooth.c</td>
<td>109</td>
</tr>
<tr>
<td>9.19</td>
<td>bluetooth/e_bluetooth.h</td>
<td>115</td>
</tr>
<tr>
<td>9.20</td>
<td>camera/fast_2_timer/e_calc.c</td>
<td>121</td>
</tr>
<tr>
<td>9.21</td>
<td>camera/slow_3_timer/e_calc.c</td>
<td>123</td>
</tr>
<tr>
<td>9.22</td>
<td>camera/fast_2_timer/e_po3030k.h</td>
<td>125</td>
</tr>
<tr>
<td>9.23</td>
<td>camera/slow_3_timer/e_po3030k.h</td>
<td>141</td>
</tr>
<tr>
<td>9.24</td>
<td>camera/fast_2_timer/e_registers.c</td>
<td>157</td>
</tr>
<tr>
<td>9.25</td>
<td>camera/slow_3_timer/e_registers.c</td>
<td>173</td>
</tr>
<tr>
<td>9.26</td>
<td>camera/fast_2_timer/e_timers.c</td>
<td>189</td>
</tr>
<tr>
<td>9.27</td>
<td>camera/slow_3_timer/e_timers.c</td>
<td>192</td>
</tr>
<tr>
<td>9.28</td>
<td>codec/e_common.inc</td>
<td>196</td>
</tr>
<tr>
<td>9.29</td>
<td>codec/e_sound.c</td>
<td>197</td>
</tr>
<tr>
<td>9.30</td>
<td>codec/e_sound.h</td>
<td>199</td>
</tr>
<tr>
<td>9.31</td>
<td>contrib/LIS_sensors_turret/e_devantech.c</td>
<td>201</td>
</tr>
<tr>
<td>9.32</td>
<td>contrib/LIS_sensors_turret/e_devantech.h</td>
<td>202</td>
</tr>
<tr>
<td>9.33</td>
<td>contrib/LIS_sensors_turret/e_sensext.c</td>
<td>204</td>
</tr>
<tr>
<td>9.34</td>
<td>contrib/LIS_sensors_turret/e_sensext.h</td>
<td>205</td>
</tr>
<tr>
<td>9.35</td>
<td>contrib/LIS_sensors_turret/e_sharp.c</td>
<td>206</td>
</tr>
<tr>
<td>9.36</td>
<td>contrib/LIS_sensors_turret/e_sharp.h</td>
<td>207</td>
</tr>
<tr>
<td>9.37</td>
<td>contrib/SWIS_com_module/ComModule.c</td>
<td>209</td>
</tr>
<tr>
<td>9.38</td>
<td>contrib/SWIS_com_module/ComModule.h</td>
<td>212</td>
</tr>
<tr>
<td>9.39</td>
<td>fft/e_fft.c</td>
<td>214</td>
</tr>
<tr>
<td>9.40</td>
<td>fft/e_fft.h</td>
<td>215</td>
</tr>
<tr>
<td>9.41</td>
<td>fft/e_fft_utilities.h</td>
<td>216</td>
</tr>
<tr>
<td>9.42</td>
<td>fft/e_input_signal.c</td>
<td>217</td>
</tr>
<tr>
<td>9.43</td>
<td>fft/e_twiddle_factors.c</td>
<td>218</td>
</tr>
<tr>
<td>9.44</td>
<td>I2C/e_I2C_master_module.c</td>
<td>219</td>
</tr>
</tbody>
</table>
Chapter 1

e-puck standard library documentation

1.1 Introduction

This project has been started at the Ecole Polytechnique Federale de Lausanne as collaboration between the Autonomous Systems Lab, the Swarm-Intelligent Systems group and the Laboratory of Intelligent System. An educational robot: The main goal of this project is to develop a miniature mobile robot for educational purposes at university level. To achieve this goal the robot needs, in our opinion, the following features:

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

1.2 Documentation organization

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

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

- http://www.e-puck.org/ The official site of the e-puck
- https://gna.org/projects/e-puck/ The developers area at gna
- http://lsro.epfl.ch/ The site of the lab where the e-puck has been created
Chapter 2

e-puck Module Index

2.1 e-puck Modules

Here is a list of all modules:

Analogic/Digital conversion (ADC) .................................................. 11
Bluetooth .................................................................................. 14
Camera fast two timers ............................................................... 15
Camera slow three timers ............................................................ 18
Sound ..................................................................................... 21
LIS sensor turret .................................................................... 23
Radio communication .............................................................. 24
FFT ......................................................................................... 25
I2C ......................................................................................... 27
Matlab communication ............................................................. 31
Ports, motors and LEDs ............................................................ 32
UART ....................................................................................... 36
Chapter 3

e-puck Directory Hierarchy

3.1 e-puck Directories

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

<table>
<thead>
<tr>
<th>Directory</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_d</td>
<td>37</td>
</tr>
<tr>
<td>advance_ad_scan</td>
<td>38</td>
</tr>
<tr>
<td>bluetooth</td>
<td>40</td>
</tr>
<tr>
<td>camera</td>
<td>41</td>
</tr>
<tr>
<td>fast_2_timer</td>
<td>44</td>
</tr>
<tr>
<td>slow_3_timer</td>
<td>52</td>
</tr>
<tr>
<td>codec</td>
<td>42</td>
</tr>
<tr>
<td>contrib</td>
<td>43</td>
</tr>
<tr>
<td>LIS_sensors_turret</td>
<td>48</td>
</tr>
<tr>
<td>SWIS_com_module</td>
<td>53</td>
</tr>
<tr>
<td>fft</td>
<td>46</td>
</tr>
<tr>
<td>I2C</td>
<td>47</td>
</tr>
<tr>
<td>matlab</td>
<td>49</td>
</tr>
<tr>
<td>matlab_files</td>
<td>50</td>
</tr>
<tr>
<td>motor_led</td>
<td>51</td>
</tr>
<tr>
<td>advance_one_timer</td>
<td>39</td>
</tr>
<tr>
<td>fast_agenda</td>
<td>45</td>
</tr>
<tr>
<td>uart</td>
<td>54</td>
</tr>
</tbody>
</table>
Chapter 4

e-puck Data Structure Index

4.1 e-puck Data Structures

Here are the data structures with brief descriptions:

- **AgendaList** (Manage the different agendas lists) ................................. 55
- **AgendaType** (Struct Agenda as chained list) ................................. 57
- **BtDevice** (General struct for Bluetooth device) ................................. 59
- **BtEPuck** (General struct for other e-puck) ................................. 60
- **TypeAccRaw** (Struct to store the acceleration raw data in Cartesian coord) ................................. 61
- **TypeAccSpheric** (Struct to store the acceleration vector in spherical coord) ................................. 62
Chapter 5

e-puck File Index

5.1 e-puck File List

Here is a list of all files with brief descriptions:

- `a_d/e_accelerometer.c` (Accessing the accelerometer sensor data) ........................................ 102
- `a_d/e_accelerometer.h` (Accessing the accelerometer sensor data) ........................................ 104
- `a_d/e_ad_conv.c` (Module for the Analogic/Digital conversion) ............................................. 77
- `a_d/e_ad_conv.h` (Module for the Analogic/Digital conversion) ............................................. 81
- `a_d/e_micro.c` (Accessing the microphone data) ........................................................................ 84
- `a_d/e_micro.h` (Accessing the microphone data) ........................................................................ 88
- `a_d/e_prox.c` (Accessing proximity sensor of e-puck with timer 1) .......................................... 93
- `a_d/e_prox.h` (Accessing proximity sensor of e-puck with timer 1) .......................................... 99
- `a_d/e_prox_timer2.c` (Control proximity sensor of e-puck with timer2) ................................... 106
- `a_d/advance_ad_scan/e_acc.c` (Accessing the accelerometer data) ......................................... 63
- `a_d/advance_ad_scan/e_acc.h` (Accessing the accelerometer data) ........................................... 69
- `a_d/advance_ad_scan/e_ad_conv.c` (Module for the advance Analogic/Digital conversion) ....... 74
- `a_d/advance_ad_scan/e_ad_conv.h` (Module for the advance Analogic/Digital conversion) ....... 78
- `a_d/advance_ad_scan/e_micro.c` (Accessing the microphone data) ........................................... 82
- `a_d/advance_ad_scan/e_micro.h` (Accessing the microphone data) ........................................... 86
- `a_d/advance_ad_scan/e_prox.c` (Accessing proximity sensor of e-puck) .................................... 90
- `a_d/advance_ad_scan/e_prox.h` (Accessing proximity sensor of e-puck) .................................... 96
- `bluetooth/e_bluetooth.c` (Manage Bluetooth) .......................................................................... 109
- `bluetooth/e_bluetooth.h` (Manage Bluetooth) ........................................................................... 115
- `camera/fast_2_timer/e_calc.c` (Calculate the timing for the camera (two timers)) ................... 121
- `camera/fast_2_timer/e_po3030k.h` (PO3030k library header (two timers)) ............................. 125
- `camera/fast_2_timer/e_registers.c` (Manage po3030k registers (two timers)) .................... 157
- `camera/fast_2_timer/e_timers.c` (Manage camera’s interrupts (two timers)) ......................... 189
- `camera/slow_3_timer/e_calc.c` (Calculate the timing for the camera (three timers)) ............ 123
- `camera/slow_3_timer/e_po3030k.h` (PO3030k library header (three timers)) ....................... 141
- `camera/slow_3_timer/e_registers.c` (Manage po3030k registers (three timers)) .................. 173
- `camera/slow_3_timer/e_timers.c` (Manage camera’s interrupts (three timers)) .................... 192
- `codec/e_common.inc` ........................................................................................................... 196
- `codec/e_sound.c` (Package to play basics sounds on the e-puck’s speaker. For more info look at this: `Sound` (p. 21)) ................................................................. 197
- `codec/e_sound.h` (Package to play basics sounds on the e-puck’s speaker) .......................... 199
- `contrib/LIS_sensors_turret/e_devantech.c` ........................................................................... 201
- `contrib/LIS_sensors_turret/e_devantech.h` (Devantech sensor of e-puck) ......................... 202
<table>
<thead>
<tr>
<th>File Path</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>contrib/LIS_sensors_turret/e_sensext.c</td>
<td>(Radio communication)</td>
<td>204</td>
</tr>
<tr>
<td>contrib/LIS_sensors_turret/e_sensext.h</td>
<td>(Radio communication)</td>
<td>205</td>
</tr>
<tr>
<td>contrib/LIS_sensors_turret/e_sharp.c</td>
<td>(Radio communication)</td>
<td>206</td>
</tr>
<tr>
<td>contrib/LIS_sensors_turret/e_sharp.h</td>
<td>(Radio communication)</td>
<td>207</td>
</tr>
<tr>
<td>contrib/SWIS_com_module/ComModule.c</td>
<td>(Radio communication)</td>
<td>209</td>
</tr>
<tr>
<td>contrib/SWIS_com_module/ComModule.h</td>
<td>(Radio communication)</td>
<td>212</td>
</tr>
<tr>
<td>matplotlib/matlab files/EpuckSendData.m</td>
<td>(Radio communication)</td>
<td>214</td>
</tr>
<tr>
<td>matplotlib/matlab files/CloseEpuck.m</td>
<td>(Radio communication)</td>
<td>215</td>
</tr>
<tr>
<td>matplotlib/matlab files/EpuckFlush.m</td>
<td>(Radio communication)</td>
<td>216</td>
</tr>
<tr>
<td>matplotlib/matlab files/EpuckGetData.m</td>
<td>(Radio communication)</td>
<td>217</td>
</tr>
<tr>
<td>matplotlib/matlab files/EpuckSendData.m</td>
<td>(Radio communication)</td>
<td>218</td>
</tr>
<tr>
<td>I2C/e_I2C_master_module.c</td>
<td>(Manage I2C basics)</td>
<td>219</td>
</tr>
<tr>
<td>I2C/e_I2C_master_module.h</td>
<td>(Manage I2C basics)</td>
<td>223</td>
</tr>
<tr>
<td>I2C/e_I2C_protocol.c</td>
<td>(Manage I2C protocol)</td>
<td>227</td>
</tr>
<tr>
<td>I2C/e_I2C_protocol.h</td>
<td>(Manage I2C protocol)</td>
<td>229</td>
</tr>
<tr>
<td>Matlab/matlab.c</td>
<td>(To communicate with Matlab)</td>
<td>233</td>
</tr>
<tr>
<td>Matlab/matlab.h</td>
<td>(To communicate with Matlab)</td>
<td>237</td>
</tr>
<tr>
<td>Matlab/matlab files/OpenEpuck.m</td>
<td>(To communicate with Matlab)</td>
<td>239</td>
</tr>
<tr>
<td>motor_led/e_epuck_ports.h</td>
<td>(Define all the usefull names corresponding of e-puck’s hardware)</td>
<td>232</td>
</tr>
<tr>
<td>motor_led/e_init_port.c</td>
<td>(Initialize the ports on standard configuration)</td>
<td>237</td>
</tr>
<tr>
<td>motor_led/e_init_port.h</td>
<td>(Initialize the ports on standard configuration)</td>
<td>238</td>
</tr>
<tr>
<td>motor_led/e_led.c</td>
<td>(Manage the LEDs. A little example for the LEDs (all the LEDs are blinking)</td>
<td>261</td>
</tr>
<tr>
<td>motor_led/e_led.h</td>
<td>(Manage the LEDs. A little example for the LEDs (all the LEDs are blinking)</td>
<td>274</td>
</tr>
<tr>
<td>motor_led/e_motors.c</td>
<td>(Manage the motors (with timer 4 and 5))</td>
<td>284</td>
</tr>
<tr>
<td>motor_led/e_motors.h</td>
<td>(Manage the motors (with timer 4 and 5))</td>
<td>297</td>
</tr>
<tr>
<td>motor_led/e_motors_timer3.c</td>
<td>(Initialize the ports on standard configuration)</td>
<td>329</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_agenda.c</td>
<td>(Manage the agendas (timer2))</td>
<td>241</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_agenda.h</td>
<td>(Manage the agendas (timer2))</td>
<td>245</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_led.c</td>
<td>(Manage the LEDs with blinking possibility (timer2))</td>
<td>250</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_led.h</td>
<td>(Manage the LEDs with blinking possibility (timer2))</td>
<td>263</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_motors.c</td>
<td>(Manage the motors (with timer2))</td>
<td>276</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_motors.h</td>
<td>(Manage the motors (with timer2))</td>
<td>287</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_motors_kml.c</td>
<td>(Manage the motors (with timer2))</td>
<td>301</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_remote_control.c</td>
<td>(Manage the IR receiver module (timer2))</td>
<td>306</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/e_remote_control.h</td>
<td>(Manage the LEDs with blinking possibility (timer2))</td>
<td>309</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/fast_agenda/e_agenda_fast.c</td>
<td>(Manage the fast agendas (timer1, 2, 3))</td>
<td>313</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/fast_agenda/e_agenda_fast.h</td>
<td>(Manage the fast agendas (timer1, 2, 3))</td>
<td>318</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/fast_agenda/e_led.c</td>
<td>(Manage the LEDs with blinking possibility (timer1, 2, 3))</td>
<td>256</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/fast_agenda/e_led.h</td>
<td>(Manage the LEDs with blinking possibility (timer1, 2, 3))</td>
<td>259</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/fast_agenda/e_motors.c</td>
<td>(Manage the motors (with timer1, 2, 3))</td>
<td>280</td>
</tr>
<tr>
<td>motor_led/advance_one_timer/fast_agenda/e_motors.h</td>
<td>(Manage the motors (with timer1, 2, 3))</td>
<td>292</td>
</tr>
<tr>
<td>uart/e_epuck_ports.inc</td>
<td>(Manage UART)</td>
<td>333</td>
</tr>
<tr>
<td>uart/e_uart_char.h</td>
<td>(Manage UART)</td>
<td>334</td>
</tr>
</tbody>
</table>

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
Chapter 6

e-puck Module Documentation

6.1 Analogic/Digital conversion (ADC)

Files

- file e_acc.c
  Accessing the accelerometer data.

- file e_acc.h
  Accessing the accelerometer data.

- file e_ad_conv.c
  Module for the advance Analogic/Digital conversion.

- file e_ad_conv.h
  Module for the advance Analogic/Digital conversion.

- file e_micro.c
  Accessing the microphone data.

- file e_micro.h
  Accessing the microphone data.

- file e_prox.c
  Accessing proximity sensor of e-puck.

- file e_prox.h
  Accessing proximity sensor of e-puck.

- file e_accelerometer.c
  Accessing the accelerometer sensor data.

- file e_accelerometer.h
  Accessing the accelerometer sensor data.
• file \texttt{e\_ad\_conv.c}
  \smallskip
  \textit{Module for the Analogic/Digital conversion.}

• file \texttt{e\_ad\_conv.h}
  \smallskip
  \textit{Module for the Analogic/Digital conversion.}

• file \texttt{e\_micro.c}
  \smallskip
  Accessing the microphone data.

• file \texttt{e\_micro.h}
  \smallskip
  Accessing the microphone data.

• file \texttt{e\_prox.c}
  \smallskip
  Accessing proximity sensor of e-puck with timer 1.

• file \texttt{e\_prox.h}
  \smallskip
  Accessing proximity sensor of e-puck with timer 1.

• file \texttt{e\_prox\_timer2.c}
  \smallskip
  Control proximity sensor of e-puck with timer2.

\section*{6.1.1 Detailed Description}

\section*{6.1.2 Introduction}

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

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

\begin{itemize}
  \item 1 3D accelerometer
  \item 8 proximity sensors
  \item 3 microphones
\end{itemize}

\section*{6.1.3 Package organization}

This package is divided by two sub packages:

\begin{itemize}
  \item The standard approach (files located in the "a\_d" folder). This approach uses the timer1 (or timer2) to coordinate the ADC register acquisition.
  \item The more advanced approach (files located in the "a\_d/advance\_ad\_scan" folder) uses the ADC interrupt to update the four arrays containing all the data of all the peripherals which use the ADC (\texttt{e\_mic\_scan} (p. 83) for the microphones, \texttt{e\_acc\_scan} (p. 76) for the accelerometer, \texttt{e\_ambient\_ir} (p. 92) and \texttt{e\_ambient\_and\_reflected\_ir} (p. 92) for the proximity sensors). In this approach, the acquisition is made automatically and always with the same delay. The functions specific to each module (\texttt{advance\_ad\_scan/e\_acc.c} (p. 63), \texttt{advance\_ad\_scan/e\_prox.c} (p. 90), \texttt{advance\_ad\_scan/e\_micro.c} (p. 82)) are also more elaborates than the same one in the standard package.
\end{itemize}
Author:

Doc: Jonathan Besuchet
6.2 Bluetooth

Files

- file e_bluetooth.c
  Manage Bluetooth.

- file e_bluetooth.h
  Manage Bluetooth.

6.2.1 Detailed Description

6.2.2 Introduction

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

The bluetooth device is connected on the uart1.

Generally, the bluetooth modules have a bluetooth class device which identify them as PC, mobile, mouse, ...

The e-puck bluetooth module has the default device class number, it’s 000.

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

Author:

  Doc: Jonathan Besuchet
6.3 Camera fast two timers

Files

- file e_calc.c
  
  *Calculate the timing for the camera (two timers).*

- file e_po3030k.h

  *PO3030k library header (two timers).*

- file e_timers.c

  *Manage camera's interrupts (two timers).*

6.3.1 Detailed Description

6.3.2 Introduction

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

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

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

6.3.3 Default settings

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

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

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

6.3.4 Performances

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

- Size: 640x480, Subsampling: 16x16, RGB565: 4.3 fps
- Size: 16x480, Subsampling: 16x16, RGB565: 4.3 fps
- Size: 480x16, Subsampling: 16x16: RGB565: 4.3fps
- Size: 64x64, Subsampling: 4x4, RGB565: 4.3 fps
6.3.5 Important note
This driver is extremely 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 those two instructions when you use the camera. This include the _ISRFAST C macro. If you use them, some random and really hard to debug behavior will happen. You have been warned!

6.3.6 Examples

6.3.6.1 Basic example

```c
#include "e_po3030k.h"

char buffer[2*40*40];
int main(void) {
    e_po3030k_init_cam();
    e_po3030k_config_cam((ARRAY_WIDTH -160)/2,(ARRAY_HEIGHT-160)/2, 160,160,4,4,RGB_565_MODE);
    e_po3030k_write_cam_registers();
    e_po3030k_launch_capture(buffer);
    while(!e_po3030k_is_img_ready());
    // buffer contain a 40*40 RGB picture now
    ( insert usefull code here )
    return 0;
}
```

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

6.3.6.2 More advanced example

```c
#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,
```

```c
```
320, 8, 2, 4, GREY_SCALE_MODE;
    e_po3030k_set_mirror(1, 1);
    e_po3030ke_set_ref_exposure(100);
    e_po3030k_write_cam_registers();
    e_po3030k_launch_capture(buffer);
    while (!e_po3030k_is_img_ready());

    // Here buffer contain a 160x2 greyscale picture
    return 0;
}

This example configure the camera to acquire a 320x8 pixel picture, but subsampled 2x in width and 4x in height, thus resulting in a 160x2 linear greyscale picture. It "emulate" a linear camera. This example tell the camera to enable the vertical and horizontal mirror, and to set the average exposure to 100.
6.4 Camera slow three timers

Files

- file e_calc.c
  
  *Calculate the timing for the camera (three timers).*

- file e_po3030k.h
  
  *PO3030k library header (three timers).*

- file e_registers.c
  
  *Manage po3030k registers (three timers).*

- file e_timers.c
  
  *Manage camera’s interrupts (three timers).*

6.4.1 Detailed Description

Warning:

This driver version is completely interrupt driven to synchronize with the camera. This slows down the acquisition a lot. You should use the other driver’s version which synchronize with the camera only at each row.

6.4.2 Introduction

This driver expose most of the Po3030k camera interfaces. Some functions are useful, 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 kept in memory. The configuration function only alter this array. When you call, for example `e_po3030k_config_cam()` (p. 121), nothing is written on the camera, but only in the internal register representation.

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

6.4.3 Default settings

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

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

There is no default setting for the image size and color.
6.4 Camera slow three timers

6.4.4 Performances

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

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

6.4.5 Exemples

6.4.5.1 Basic example

```c
#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 40x40 RGB picture now
    // insert useful code here
    return 0;
}
```

This exemple tell de driver to acquire 160x160 pixel picture from the camera 4x subsampling, thus resulting with a 40x40 pixel. The buffer as a size of 40x40x2 because RGB565 is a two bytes per pixel data format.

6.4.5.2 More advanced example

```c
#include "e_po3030k.h"
char buffer[160*2];
int main(void) {
    e_po3030k_init_cam();
    e_po3030k_config_cam((ARRAY_WIDTH - 320)/2,(ARRAY_HEIGHT - 32)/2, 320,8,2,4,GREY_SCALE_MODE);
    e_po3030k_set_mirror(1,1);
    e_po3030k_set_ref_exposure(100);
}
```
e_po3030k_write_cam_registers();
e_po3030k_launch_capture(buffer);
while(!e_po3030k_is_img_ready());

// Here buffer contain a 160x2 greyscale picture
return 0;
}

This exemple configure the camera to acquire a 320x8 pixel picture, but subsampled 2x in width and 4x in
heigth, thus resulting in a 160x2 linear greyscale picture. It "emulate" a linear camera. This exemple tell
the camera to to enable the vertical and horizontal mirror, and to set the average exposure to 100.
6.5 Sound

Files

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

- file e_sound.h
  
  Package to play basics sounds on the e-puck’s speaker.

6.5.1 Detailed Description

6.5.2 Introduction

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

6.5.3 Package organization

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

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

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

- The "sound" which is in the file codec/e_const_sound.s is placed somewhere in e-puck’s memory.

- When you call the function e_play_sound(int sound_offset, int sound_length) (p. 200), the words are sent to the DCI one by one from the offset you have specified with the first parameter. The number of words sent are specified with the second parameter.

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

Warning:

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

6.5.4 Sounds plage

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

0, 2112 : "haa"
Then if you want to play the "yaouh" sound from the beginning to the end just write this: `e_play_sound(7294, 3732);

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

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

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

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet
6.6 LIS sensor turret

Files

- file `e_devantech.h`
  
  Devantech sensor of e-puck.

6.6.1 Detailed Description

6.6.2 Introduction

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

Files

• file ComModule.h

Radio communication.

6.7.1 Detailed Description

6.7.2 Introduction

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

Files

- file e_fft.c
  Package to manage the FFT.

- file e_fft.h
  Package to manage the FFT.

- file e_fft_utilities.h
  Some fft features.

- file e_input_signal.c
  Allocate memory and initialize the sigCmpx array.

- file e_twiddle_factors.c
  The FFT factor from Microchip.

6.8.1 Detailed Description

6.8.2 Introduction

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

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

6.8.3 How it works

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

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

Then you just have

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

A little code to illustrate this.

```c
 e_ad_scan_on();
 // waiting all the 512 data (here we scan the microphones)
 while(!e_ad_is_array_filled());
 e_ad_scan_off();
```
// We put the mean to zero
e_subtract_mean(e_mic_scan[0], FFT_BLOCK_LENGTH, LOG2_BLOCK_LENGTH);
// We copy the array of micro zero to the FFT array
e_fast_copy(e_mic_scan[0], (int*)sigCmpx, FFT_BLOCK_LENGTH);

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

Author:

Doc: Jonathan Besuchet
6.9 I2C

Files

- file e_I2C_master_module.c
  
  Manage I2C basics.

- file e_I2C_master_module.h
  
  Manage I2C basics.

- file e_I2C_protocol.c
  
  Manage I2C protocole.

- file e_I2C_protocol.h
  
  Manage I2C protocole.

6.9.1 Detailed Description

6.9.2 Introduction

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

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

- initializing the I2C on the microcontroller (**e_i2c_init(void)** (p. 225))

- sending the Start bit (**e_i2c_start(void)** (p. 225))

- sending the Restart bit (**e_i2c_restart(void)** (p. 225))

- sending the Stop bit (**e_i2c_stop(void)** (p. 226))

- sending the acknowledgement bit (**e_i2c_ack(void)** (p. 224))

- writing or receiving a byte (**e_i2c_write(char byte)** (p. 226), **e_i2c_read(char *buf)** (p. 225))

- ...

6.9.3 Overview of I2C protocol

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

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

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

1) Suppose microcontroller A wants to send information to module B:
   - microcontroller A (master), addresses module B (slave)
   - microcontroller A (master-transmitter), sends data to module B (slave-receiver)
   - microcontroller A terminates the transfer

2) If microcontroller A wants to receive information from module B:
   - microcontroller A (master) addresses module B (slave)
   - microcontroller A (master-receiver) receives data from module B (slave-transmitter)
   - microcontroller A terminates the transfer.

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

6.9.3.2 Start and Stop conditions

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

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

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

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

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

6.9.3.3 Transferring data

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

Acknowledgement:

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

The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse. Of course, set-up and hold times must also be taken into account.
Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received.

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

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

### 6.9.4 Use I2C on e-puck

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

#### 6.9.4.1 Basics functions

The functions of the files `e_I2C_master_module.c` (p. 219) and `e_I2C_master_module.h` (p. 223) are low level I2C functions. They are made to perform the basics tasks like:

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

#### 6.9.4.2 More developed functions

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

### 6.9.5 Reference

For more information about I2C:

Author:

Doc: Jonathan Besuchet
6.10 Matlab communication

Files

- file `matlab.c`
  
  To communicate with matlab.

- file `matlab.h`
  
  To communicate with matlab.

6.10.1 Detailed Description

6.10.2 Introduction

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

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

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

6.10.2.1 Sending data from matlab

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

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

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

6.10.2.2 Sending data from e-puck

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

See also:

Matlab communication (p. 31)

Author:

Doc: Jonathan Besuchet
6.11 Ports, motors and LEDs

Files

- file `e_agenda.c`
  Manage the agendas (timer2).

- file `e_agenda.h`
  Manage the agendas (timer2).

- file `e_led.c`
  Manage the LEDs with blinking possibility (timer2).

- file `e_led.h`
  Manage the LEDs with blinking possibility (timer2).

- file `e_motors.c`
  Manage the motors (with timer2).

- file `e_motors.h`
  Manage the motors (with timer2).

- file `e_motors_kml.c`
  Manage the motors (with timer2).

- file `e_remote_control.c`
  Manage the IR receiver module (timer2).

- file `e_remote_control.h`
  Manage the LEDs with blinking possibility (timer2).

- file `e_remote_control.h`
  Manage the LEDs with blinking possibility (timer2).

- file `e_agenda_fast.c`
  Manage the fast agendas (timer1, 2, 3).

- file `e_agenda_fast.h`
  Manage the fast agendas (timer1, 2, 3).

- file `e_led.c`
  Manage the LEDs with blinking possibility (timer1, 2, 3).

- file `e_led.h`
  Manage the LEDs with blinking possibility (timer1, 2, 3).

- file `e_motors.c`
  Manage the motors (with timer1, 2, 3).

- file `e_motors.h`
Manage the motors (with timer 1, 2, 3).

- file e_epuck_ports.h
  Define all the usefull names corresponding of e-puck’s hardware.

- file e_init_port.c
  Initialize the ports on standard configuration.

- file e_init_port.h
  Initialize the ports on standard configuration.

- file e_led.c
  Manage the LEDs.
  A little exemple for the LEDs (all the LEDs are blinking).

- file e_led.h
  Manage the LEDs.
  A little exemple for the LEDs (all the LEDs are blinking).

- file e_motors.c
  Manage the motors (with timer 4 and 5).

- file e_motors.h
  Manage the motors (with timer 4 and 5).

- file e_motors_timer3.c
  Initialize the ports on standard configuration.

- file e_motors_timer3.h
  Initialize the ports on standard configuration.

### 6.11.1 Detailed Description

### 6.11.2 Introduction

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

#### 6.11.2.1 Ports

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

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

#### 6.11.2.2 Motors

The e-puck has two step by step motors called MOTOR1 (left) and MOTOR2 (right). To control the changing phase’s sequence of these motors we need to use timers. Four possibilities are offered to you:
• standard: we use the timer4 for MOTOR2 and timer5 for MOTOR1. This solution is exploited by the file library\motor_led\e_motors.c.

• one timer standard: we use the timer3 for both MOTOR1 and MOTOR2. This solution is exploited by the file library\motor_led\e_motors_timer3.c

• advance one timer: we use the timer2 for both MOTOR1 and MOTOR2, but this time the mechanism work on the agenda method (see below or e_agenda.h (p. 245) for more information about agenda). This solution is exploited by the file library\motor_led\advance_one_timer\e_motors.c.

• fast agenda: we use the timer1,2,3 for both MOTOR1 and MOTOR2, but this time the mechanism work on the fast_agenda method (see below or e_agenda_fast.h (p. 318) for more information about fast_agenda). This solution is exploited by the file library\motor_led\advance_one_timer\fast_agenda\e_motors.c.

6.11.2.3 LED

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

6.11.2.4 IR remote

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

Warning:

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

6.11.3 Timer’s problems

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

Warning:

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

• one timer standard
• advance one timer
• fast agenda

6.11.4 Agenda solution

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

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

How it works?
You create an agenda by specifying:

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

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

See also:

- `e_agenda.c` (p. 241), `e_agenda_fast.c` (p. 313)

Author:

Doc: Jonathan Besuchet
6.12 UART

Files

- file e_uart_char.h
  Manage UART.

6.12.1 Detailed Description

6.12.2 Introduction

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

Warning:

In this package, the functions are written in ASM. We have "e_init_uartX.s" file for the initializing functions, "e_uartX_rx.s" file for receiving data functions and "e_uartX_tx.s" file for transmitting data functions (X can be 1 or 2).

Even these files are written in ASM, you can call them by including the e_uart_char.h file in your C code (look at the example in e_uart_char.h).

6.12.3 Bluetooth

The e-puck has its Bluetooth module connected on the uart1. Two ways are possible when you work with Bluetooth:

- you are the master, look at this: Bluetooth
- you are the slave.

When you are the slave, you can communicate with the master device exactly in the same way as you do to communicate through the uart. The Bluetooth protocol is made to look like as transparent as possible. This is possible because the master initializes the communication and the Bluetooth module answers automatically to create the connection. After that the connection was created, you can communicate with the master by using the uart protocol.

Author:

Doc: Jonathan Besuchet
Chapter 7

e-puck Directory Documentation

7.1 a_d/ Directory Reference

Directories

- directory advance_ad_scan

Files

- file e_accelerometer.c
  Accessing the accelerometer sensor data.

- file e_accelerometer.h
  Accessing the accelerometer sensor data.

- file e_ad_conv.c
  Module for the Analogic/Digital conversion.

- file e_ad_conv.h
  Module for the Analogic/Digital conversion.

- file e_micro.c
  Accessing the microphone data.

- file e_micro.h
  Accessing the microphone data.

- file e_prox.c
  Accessing proximity sensor of e-puck with timer 1.

- file e_prox.h
  Accessing proximity sensor of e-puck with timer 1.

- file e_prox_timer2.c
  Control proximity sensor of e-puck with timer 2.
7.2  a_d/advance_ad_scan/ Directory Reference

Files

- file e_acc.c
  Accessing the accelerometer data.

- file e_acc.h
  Accessing the accelerometer data.

- file e_ad_conv.c
  Module for the advance Analogic/Digital conversion.

- file e_ad_conv.h
  Module for the advance Analogic/Digital conversion.

- file e_micro.c
  Accessing the microphone data.

- file e_micro.h
  Accessing the microphone data.

- file e_prox.c
  Accessing proximity sensor of e-puck.

- file e_prox.h
  Accessing proximity sensor of e-puck.
7.3 motorLed/advance_one_timer/ Directory Reference

Directories

• directory fast_agenda

Files

• file e_agenda.c
  Manage the agendas (timer2).

• file e_agenda.h
  Manage the agendas (timer2).

• file e Led.c
  Manage the LEDs with blinking possibility (timer2).

• file e Led.h
  Manage the LEDs with blinking possibility (timer2).

• file e_motors.c
  Manage the motors (with timer2).

• file e_motors.h
  Manage the motors (with timer2).

• file e_motors_kml.c
  Manage the motors (with timer2).

• file e_remote_control.c
  Manage the IR receiver module (timer2).

• file e_remote_control.h
  Manage the LEDs with blinking possibility (timer2).
7.4 bluetooth/ Directory Reference

Files

- file e_bluetooth.c
  
  Manage Bluetooth.

- file e_bluetooth.h
  
  Manage Bluetooth.
7.5 camera/ Directory Reference

Directories

• directory fast_2_timer
• directory slow_3_timer
7.6 codec/ Directory Reference

Files

- file e_common.inc
- file e_sound.c
  Package to play basics sounds on the e-puck’s speaker.
  For more info look at this: Sound (p. 21).

- file e_sound.h
  Package to play basics sounds on the e-puck’s speaker.
7.7 contrib/ Directory Reference

Directories

- directory LIS_sensors_turret
- directory SWIS_com_module
7.8 camera/fast_2_timer/ Directory Reference

Files

- file e_calc.c
  
  Calculate the timing for the camera (two timers).

- file e_po3030k.h
  
  PO3030k library header (two timers).

- file e_registers.c
  
  Manage po3030k registers (two timers).

- file e_timers.c
  
  Manage camera’s interrupts (two timers).
Files

- **file e_agenda_fast.c**
  Manage the fast agendas (timer1, 2, 3).

- **file e_agenda_fast.h**
  Manage the fast agendas (timer1, 2, 3).

- **file e_led.c**
  Manage the LEDs with blinking possibility (timer1, 2, 3).

- **file e_led.h**
  Manage the LEDs with blinking possibility (timer1, 2, 3).

- **file e_motors.c**
  Manage the motors (with timer1, 2, 3).

- **file e_motors.h**
  Manage the motors (with timer1, 2, 3).
7.10 fft/ Directory Reference

Files

- file e_fft.c
  Package to manage the FFT.

- file e_fft.h
  Package to manage the FFT.

- file e_fft_utilities.h
  Some fft features.

- file e_input_signal.c
  Allocate memory and initialize the sigCmpx array.

- file e_twiddle_factors.c
  The FFT factor from Microchip.
7.11 I2C/ Directory Reference

Files

• file e_I2C_master_module.c
  Manage I2C basics.

• file e_I2C_master_module.h
  Manage I2C basics.

• file e_I2C_protocol.c
  Manage I2C protocol.

• file e_I2C_protocol.h
  Manage I2C protocol.
7.12 contrib/LIS_sensors_turret/ Directory Reference

Files

- file e_devantech.c
- file e_devantech.h
  
  Devantech sensor of e-puck.

- file e_sensext.c
- file e_sensext.h
- file e_sharp.c
- file e_sharp.h
7.13 matlab/ Directory Reference

Directories

- directory matlab files

Files

- file matlab.c
  To communicate with matlab.

- file matlab.h
  To communicate with matlab.
7.14 matlab/matlab files/ Directory Reference

Files

- file CloseEpuck.m
- file EpuckFlush.m
- file EpuckGetData.m
- file EpuckSendData.m
- file OpenEpuck.m
7.15 motor_led/ Directory Reference

Directories

- directory advance_one_timer

Files

- file e_epuck_ports.h
  
  Define all the useful names corresponding of e-puck’s hardware.

- file e_init_port.c
  
  Initialize the ports on standard configuration.

- file e_init_port.h
  
  Initialize the ports on standard configuration.

- file e_led.c
  
  Manage the LEDs.
  A little example for the LEDs (all the LEDs are blinking).

- file e_led.h
  
  Manage the LEDs.
  A little example for the LEDs (all the LEDs are blinking).

- file e_motors.c
  
  Manage the motors (with timer 4 and 5).

- file e_motors.h
  
  Manage the motors (with timer 4 and 5).

- file e_motors_timer3.c
  
  Initialize the ports on standard configuration.
7.16 camera/slow_3_timer/ Directory Reference

Files

- file e_calc.c
  
  Calculate the timing for the camera (three timers).

- file e_po3030k.h
  
  PO3030k library header (three timers).

- file eRegisters.c
  
  Manage po3030k registers (three timers).

- file e_timers.c
  
  Manage camera's interrupts (three timers).
7.17 contrib/SWIS_com_module/ Directory Reference

Files

- file ComModule.c
- file ComModule.h

Radio communication.
7.18 uart/ Directory Reference

Files

- file e_epuck_ports.inc
- file e_uart_char.h

Manage UART.
Chapter 8

e-puck Data Structure Documentation

8.1 AgendaList Struct Reference

Manage the different agendas lists.
#include <e_agenda_fast.h>

Data Fields

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

8.1.1 Detailed Description

Manage the different agendas lists.

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

8.1.2 Field Documentation

8.1.2.1 char AgendaList::motors

8.1.2.2 Agenda * AgendaList::waiting

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

8.1.2.3 Agenda * AgendaList::agendas[3]

We use the 3 first timers.
8.1.2.4 unsigned char AgendaList::timers_in_use[3]

Determine which one we use currently

8.1.2.5 unsigned AgendaList::speed[3]

Base speed 0.1 ms but use of multiplication

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

- motor_led/advance_one_timer/fast_agenda/e_agenda_fast.h
8.2 AgendaType Struct Reference

srtuct Agenda as chained list
#include <e_agenda.h>

Data Fields

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

8.2.1 Detailed Description

srtuct Agenda as chained list

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

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

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

8.2.2 Field Documentation

8.2.2.1 unsigned int AgendaType::cycle

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

8.2.2.2 int AgendaType::counter

count the number of interrupts

8.2.2.3 char AgendaType::activate

can be on=1 or off=0

8.2.2.4 void(* AgendaType::function)(void) (void)

function called when counter > cycle,

Warning:

This function must have the following prototype: "void func(void)"
8.2.2.5 Agenda* AgendaType::next

pointer on the next agenda

8.2.2.6 Agenda* AgendaType::next

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

- motor_led/advance_one_timer/e_agenda.h
- motor_led/advance_one_timer/fast_agenda/e_agenda_fast.h
8.3 BtDevice Struct Reference

general struct for bluetooth device
#include <e_bluetooth.h>

Data Fields

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

8.3.1 Detailed Description

general struct for bluetooth device

8.3.2 Field Documentation

8.3.2.1 unsigned char BtDevice::address[6]

Bluetooth MAC address

8.3.2.2 unsigned char BtDevice::class[3]

Bluetooth class of device

8.3.2.3 char BtDevice::friendly_name[20]

The friendly name of the bluetooth device
The documentation for this struct was generated from the following file:

- bluetooth/e_bluetooth.h
Data Structure Documentation

8.4 BtEPuck Struct Reference

general struct for other e-puck
#include <e_bluetooth.h>

Data Fields

• unsigned char address [6]
• unsigned char number [5]

8.4.1 Detailed Description

general struct for other e-puck

8.4.2 Field Documentation

8.4.2.1 unsigned char BtEPuck::address[6]

e-puck’s bluetooth MAC address

8.4.2.2 unsigned char BtEPuck::number[5]

e-puck’s bluetooth PIN
The documentation for this struct was generated from the following file:

• bluetooth/e_bluetooth.h
8.5 TypeAccRaw Struct Reference

struct to store the acceleration raw data in carthesian coord
#include <e_acc.h>

Data Fields

• int acc_x
• int acc_y
• int acc_z

8.5.1 Detailed Description

struct to store the acceleration raw data in carthesian coord

8.5.2 Field Documentation

8.5.2.1 int TypeAccRaw::acc_x

The acceleration on x axis

8.5.2.2 int TypeAccRaw::acc_y

The acceleration on y axis

8.5.2.3 int TypeAccRaw::acc_z

The acceleration on z axis

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

• a_d/advance_ad_scan/e_acc.h
8.6 TypeAccSpheric Struct Reference

struct to store the acceleration vector in spherical coord
#include <e_acc.h>

Data Fields

- float acceleration
- float orientation
- float inclination

8.6.1 Detailed Description

struct to store the acceleration vector in spherical coord

8.6.2 Field Documentation

8.6.2.1 float TypeAccSpheric::acceleration

length of the acceleration vector = intensity of the acceleration

8.6.2.2 float TypeAccSpheric::orientation

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

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

8.6.2.3 float TypeAccSpheric::inclination

inclination angle with the horizontal plan

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

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

- a_d/advance_ad_scan/e_acc.h
Chapter 9

e-puck File Documentation

9.1 a_d/advance_ad_scan/e_acc.c File Reference

Accessing the accelerometer data.
#include "math.h"
#include "e_acc.h"
#include "e_ad_conv.h"
#include "../../motor_led/e_epuck_ports.h"
#include "../../motor_led/advance_one_timer/e_led.h"
#include <stdlib.h>

Functions

• int e_get_acc (unsigned int captor)
  Read the last value of a given accelerator axes.

• int e_get_acc_filtered (unsigned int captor, unsigned int filter_size)
  Read the value of a channel, filtered by an averaging filter.

• void calculate_acc_raw (void)
  read the x, y, z values, apply an averaging filter and finally subtract the center values.

• void calculate_acc_spherical (void)
  calculate the intensity of the acceleration vector, and the Euler’s angles

• void e_acc_calibr (void)
  initialize de ad converter and calculate the zero values

• TypeAccSpheric e_read_acc_spheric (void)
  calculate and return the accel. in spherical coord

• float e_read_inclination (void)
  calculate and return the inclination angle
• float e_read_orientation (void)
  calculate and return the orientation angle

• float e_read_acc (void)
  calculate and return the intensity of the acceleration

• TypeAccRaw e_read_acc_xyz (void)
  calculate and return acceleration on the x,y,z axis

• int e_read_acc_x (void)
  calculate and return acceleration on the x axis

• int e_read_acc_y (void)
  calculate and return acceleration on the y axis

• int e_read_acc_z (void)
  calculate and return acceleration on the z axis

• void e_display_angle (void)
  light the led according to the orientation angle

Variables

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

9.1.1 Detailed Description

Accessing the accelerometer data.

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

Two structures are used:

• TypeAccSpheric (p. 62) to store the acceleration data on spherical coordinates.
• TypeAccRaw (p. 61) to store the acceleration data on cartesian coordinates.
A little exemple to read the accelerator.

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

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

**Author:**

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

### 9.1.2 Function Documentation

#### 9.1.2.1 void calculate_acc_raw (void)

read the x, y, z values, apply an averaging filter and finally subtract the center values.

#### 9.1.2.2 void calculate_acc_spherical (void)

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

#### 9.1.2.3 void e_acc_calibr (void)

initialize de ad converter and calculate the zero values

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

#### 9.1.2.4 void e_display_angle (void)

light the led according to the orientation angle
9.1.2.5 int e_get_acc (unsigned int captor)
Read the last value of a given accelerator axes.

Parameters:

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

Returns:
value filtered channel’s value

9.1.2.6 int e_get_acc_filtered (unsigned int captor, unsigned int filter_size)
Read the value of a channel, filtered by an averaging filter.

Parameters:

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

Returns:
value filtered channel’s value

9.1.2.7 float e_read_acc (void)
calculate and return the intensity of the acceleration

Returns:
intensity of the acceleration vector

9.1.2.8 TypeAccSpheric e_read_acc_spheric (void)
calculate and return the accel. in spherical coord

Returns:
acceleration in spherical coord

See also:
TypeAccSpheric (p. 62)

9.1.2.9 int e_read_acc_x (void)
calculate and return acceleration on the x axis

Returns:
acceleration on the x axis
9.1.2.10 TypeAccRaw e_read_acc_xyz (void)
calculate and return acceleration on the x,y,z axis

Returns:
acceleration on the x,y,z axis

See also:
TypeAccRaw (p. 61)

9.1.2.11 int e_read_acc_y (void)
calculate and return acceleration on the y axis

Returns:
acceleration on the y axis

9.1.2.12 int e_read_acc_z (void)
calculate and return acceleration on the z axis

Returns:
acceleration on the z axis

9.1.2.13 float e_read_inclination (void)
calculate and return the inclination angle

Returns:
inclination angle of the robot

9.1.2.14 float e_read_orientation (void)
calculate and return the orientation angle

Returns:
orientation of the accel vector
9.1.3 Variable Documentation

9.1.3.1 int acc_x [static]

9.1.3.2 int acc_y [static]

9.1.3.3 int acc_z [static]

9.1.3.4 float acceleration [static]

9.1.3.5 int angle_mem = 0 [static]

9.1.3.6 int centre_x = 0 [static]

9.1.3.7 int centre_y = 0 [static]

9.1.3.8 int centre_z = 2000 [static]

9.1.3.9 int e_acc_scan[3][ACC_SAMP_NB]

Array to store the acc values

9.1.3.10 unsigned int e_last_acc_scan_id

9.1.3.11 float inclination [static]

9.1.3.12 float orientation [static]
Accessing the accelerometer data.

**Data Structures**

- **struct** TypeAccSpheric
  
  *struct to store the acceleration vector in spherical coord*

- **struct** TypeAccRaw
  
  *struct to store the acceleration raw data in cartesian coord*

**Defines**

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

**Functions**

- int e_get_acc (unsigned int captor)
  
  *Read the last value of a given accelerator axes.*

- int e_get_acc_filtered (unsigned int captor, unsigned int filter_size)
  
  *Read the value of a channel, filtered by an averaging filter.*

- **TypeAccSpheric** e_read_acc_spheric (void)
  
  *calculate and return the accel. in spherical coord*

- float e_read_orientation (void)
  
  *calculate and return the orientation angle*

- float e_read_inclination (void)
  
  *calculate and return the inclination angle*

- float e_read_acc (void)
  
  *calculate and return the intensity of the acceleration*

- **TypeAccRaw** e_read_acc_xyz (void)
  
  *calculate and return acceleration on the x,y,z axis*

- int e_read_acc_x (void)
  
  *calculate and return acceleration on the x axis*
• int e_read_acc_y (void)
calculate and return acceleration on the y axis

• int e_read_acc_z (void)
calculate and return acceleration on the z axis

• void e_acc_calibr (void)
initialize de ad converter and calculate the zero values

• void e_display_angle (void)
light the led according to the orientation angle

9.2.1 Detailed Description

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

Two structures are used:

• TypeAccSpheric (p. 62) to store the acceleration data on spherical coordinates.
• TypeAccRaw (p. 61) to store the acceleration data on cartesian coordinates.

A little example to read the accelerometer.

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

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

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet
9.2.2 Define Documentation

9.2.2.1 #define ACCX_BUFFER 0

9.2.2.2 #define ACCY_BUFFER 1

9.2.2.3 #define ACCZ_BUFFER 2

9.2.2.4 #define ANGLE_ERROR 666.0

9.2.2.5 #define CST_RADIAN (180.0/3.1415)

9.2.2.6 #define FILTER_SIZE 5

9.2.3 Function Documentation

9.2.3.1 void e_acc_calibr (void)

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

9.2.3.2 void e_display_angle (void)

light the led according to the orientation angle

9.2.3.3 int e_get_acc (unsigned int captor)

Read the last value of a given accelerometer axes.

Parameters:

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

Returns:

  value filtered channel’s value

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

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

Parameters:

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

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

Returns:

  value filtered channel’s value
9.2.3.5 float e_read_acc (void)
calculate and return the intensity of the acceleration

**Returns:**
- intensity of the acceleration vector

9.2.3.6 TypeAccSpheric e_read_acc_spheric (void)
calculate and return the accel. in spherical coord

**Returns:**
- acceleration in spherical coord

**See also:**
- TypeAccSpheric (p. 62)

9.2.3.7 int e_read_acc_x (void)
calculate and return acceleration on the x axis

**Returns:**
- acceleration on the x axis

9.2.3.8 TypeAccRaw e_read_acc_xyz (void)
calculate and return acceleration on the x,y,z axis

**Returns:**
- acceleration on the x,y,z axis

**See also:**
- TypeAccRaw (p. 61)

9.2.3.9 int e_read_acc_y (void)
calculate and return acceleration on the y axis

**Returns:**
- acceleration on the y axis
9.2.3.10  int e_read_acc_z (void)
calculate and return acceleration on the z axis

Returns:
   acceleration on the z axis

9.2.3.11  float e_read_inclination (void)
calculate and return the inclination angle

Returns:
   inclination angle of the robot

9.2.3.12  float e_read_orientation (void)
calculate and return the orientation angle

Returns:
   orientation of the accel vector
9.3  a_d/advance_ad_scan/e_ad_conv.c File Reference

Module for the advance Analogic/Digital conversion.
#include "./../../motor_led/e_epuck_ports.h"
#include "e_ad_conv.h"

Functions

• void e_init_ad_scan (unsigned char only_micro)
  Initialize all the A/D register needed.

• void __attribute__((__interrupt__, auto_psv))
  Save the AD buffer registers in differents arrays.

• unsigned char e_ad_is_acquisition_completed (void)
  To know if the ADC acquisition is completed.

• unsigned char e_ad_is_array_filled (void)
  To know if the ADC acquisition of microphone only is completed.

• void e_ad_scan_on (void)
  Enable the ADC conversion.

• void e_ad_scan_off (void)
  Disable the ADC conversion.

Variables

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

9.3.1 Detailed Description

Module for the advance Analogic/Digital conversion.

Author:
  Code: Francesco Mondada, Michael-Bonani & Borter Jean-Joel
  Doc: Jonathan Besuchet
9.3.2 Function Documentation

9.3.2.1 void __attribute__((interrupt__, auto_psv))

Save the AD buffer registers in different arrays.
Interrupt from timer3.
Interrupt from timer2.
Parse the chained list of agenda.
Increment counter only.
Check if agenda has to be treated according to the cycle value and current counter value.
Do it for number of cycle positive or null.
Check if a service has to be activated.

9.3.2.2 unsigned char e_ad_is_acquisition_completed (void)

To know if the ADC acquisition is completed.

Returns:

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

9.3.2.3 unsigned char e_ad_is_array_filled (void)

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

Returns:

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

9.3.2.4 void e_ad_scan_off (void)

Disable the ADC conversion.

9.3.2.5 void e_ad_scan_on (void)

Enable the ADC conversion.

9.3.2.6 void e_init_ad_scan (unsigned char only_micro)

Initialize all the A/D register needed.
Set up the different ADC register to process the AD conversion by scanning the used AD channels. Each value of the channels will be stored in a different AD buffer register and an interrupt will occur at the end of the scan.

Parameters:

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

9.3.3.1 int e_acc_scan[3][ACC_SAMP_NB]
Array to store the acc values

9.3.3.2 int e_ambient_and_reflected_ir[8]
Array to store the light when IR led is on

9.3.3.3 int e_ambient_ir[8]
Array to store the ambient light measurement

9.3.3.4 unsigned int e_last_acc_scan_id = 0

9.3.3.5 unsigned int e_last_mic_scan_id = 0

9.3.3.6 int e_mic_scan[3][MIC_SAMP_NB]
Array to store the mic values

9.3.3.7 unsigned char is_ad_acquisition_completed = 0 [static]

9.3.3.8 unsigned char is_ad_array_filled = 0 [static]

9.3.3.9 unsigned char micro_only = 0 [static]
Module for the Analogic/Digital conversion.
#include "../motor_led/e_epuck_ports.h"

Functions

- void e_init_ad (void)
  
  *Initialize all the A/D register needed.*

- int e_read_ad (unsigned int channel)
  
  *Function to sample an AD channel.*

9.4.1 Detailed Description

Module for the Analogic/Digital conversion.

Author:

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

9.4.2 Function Documentation

9.4.2.1 void e_init_ad (void)

Initialize all the A/D register needed.

9.4.2.2 int e_read_ad (unsigned int channel)

Function to sample an AD channel.

Parameters:

  \textit{channel} The A/D channel you want to sample Must be between 0 to 15

Returns:

  The sampled value on the specified channel
9.5  a_d/advance_ad_scan/e_ad_conv.h File Reference

Module for the advance Analogic/Digital conversion.

Defines

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

Functions

- `void e_init_ad_scan (unsigned char only_micro)`
  
  *Initialize all the A/D register needed.*

- `unsigned char e_ad_is_array_filled (void)`
  
  *To know if the ADC acquisition of microphone only is completed.*

- `unsigned char e_ad_is_acquisition_completed (void)`
  
  *To know if the ADC acquisition is completed.*

- `void e_ad_scan_on (void)`
  
  *Enable the ADC conversion.*

- `void e_ad_scan_off (void)`
  
  *Disable the ADC conversion.*

9.5.1  Detailed Description

Module for the advance Analogic/Digital conversion.

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

The data sampled are stored in the corresponding array:

- `e_mic_scan`  (p. 83)[3][MIC_SAMP_NB] Array to store the mic values
- `e_acc_scan`  (p. 76)[3][ACC_SAMP_NB] Array to store the acc values
- `e_ambient Ir` (p. 92)[8] Array to store ambient light measurement
• e_ambient_and_reflected_ir (p. 92) Array to store ambient and reflected light measurement

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

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

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

Author:

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

9.5.2 Define Documentation

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

9.5.2.2 #define ACC_PROX_SAMP_FREQ 256.0

9.5.2.3 #define ACC_SAMP_NB 50

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

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

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

9.5.2.7 #define ALL_ADC 0

9.5.2.8 #define MIC_SAMP_FREQ 16384.0

9.5.2.9 #define MIC_SAMP_NB 100

9.5.2.10 #define MICRO_ONLY 1

9.5.2.11 #define PULSE_LENGHT 0.0003

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

9.5.3 Function Documentation

9.5.3.1 unsigned char e_ad_is_acquisition_completed (void)

To know if the ADC acquisition is completed.

Returns:

0 if the new acquisition is not made, 1 if completed.
### 9.5.3.2 unsigned char e_ad_is_array_filled (void)

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

**Returns:**

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

### 9.5.3.3 void e_ad_scan_off (void)

Disable the ADC conversion.

### 9.5.3.4 void e_ad_scan_on (void)

Enable the ADC conversion.

### 9.5.3.5 void e_init_ad_scan (unsigned char only_micro)

Initialize all the A/D register needed.

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

**Parameters:**

- **only_micro** Put MICRO_ONLY to use only the three microphones at 33kHz. Put ALL_ADC to use all the stuff using the ADC.
Module for the Analogic/Digital conversion.

**Functions**

- **void e_init_ad (void)**
  
  Initialize all the A/D register needed.

- **int e_read_ad (unsigned int channel)**
  
  Function to sample an AD channel.

**9.6.1 Detailed Description**

Module for the Analogic/Digital conversion.

**Author:**

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

**9.6.2 Function Documentation**

**9.6.2.1 void e_init_ad (void)**

Initialize all the A/D register needed.

**9.6.2.2 int e_read_ad (unsigned int channel)**

Function to sample an AD channel.

**Parameters:**

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

**Returns:**

The sampled value on the specified channel
9.7 a_d/advance_ad_scan/e_micro.c File Reference

Accessing the microphone data.

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

Functions

- int e_get_micro (unsigned int micro_id)
  Get the last value of a given micro.

- int e_get_micro_average (unsigned int micro_id, unsigned int filter_size)
  Get the average on a given number of sample from a micro.

- int e_get_micro_volume (unsigned int micro_id)
  Get the difference between the highest and lowest sample.

- void e_get_micro_last_values (int micro_id, int *result, unsigned samples_nb)
  Write to a given array, the last values of one micro.

Variables

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

9.7.1 Detailed Description

Accessing the microphone data.

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

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

9.7.2 Function Documentation

9.7.2.1 int e_get_micro (unsigned int micro_id)

Get the last value of a given micro.

Parameters:

- micro_id micro’s ID (0, 1, or 2) (use MIC0_BUFFER (p. 86), MIC1_BUFFER (p. 86), MIC2_BUFFER (p. 86) defined in e_micro.h)
Returns:
result last value of the micro

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

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

Parameters:
micro_id micro’s ID (0, 1, or 2) (use MIC0_BUFFER (p. 86), MIC1_BUFFER (p. 86), MIC2_BUFFER (p. 86) defined in e_micro.h)
filter_size number of sample to average

Returns:
result last value of the micro

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

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

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

Parameters:
micro_id micro’s ID (0, 1, or 2) (use MIC0_BUFFER (p. 86), MIC1_BUFFER (p. 86), MIC2_BUFFER (p. 86) defined in e_micro.h)
*result pointer on the result array
samples_nb size of the result array (must be between 1 and MIC_SAMP_NB (p. 79))

9.7.2.4 int e_get_micro_volume (unsigned int micro_id)

Get the difference between the highest and lowest sample.

Parameters:
micro_id micro’s ID (0, 1, or 2) (use MIC0_BUFFER (p. 86), MIC1_BUFFER (p. 86), MIC2_BUFFER (p. 86) defined in e_micro.h)

Returns:
result volume

9.7.3 Variable Documentation

9.7.3.1 unsigned int e_last_mic_scan_id

9.7.3.2 int e_mic_scan[3][MIC_SAMP_NB]

Array to store the mic values

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
9.8  a_d/e_micro.c File Reference

Accessing the microphone data.

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

Functions

• void e_init_micro(void)
  Init the microphone A/D converter.

• void e_get_micro(int *m0, int *m1, int *m2)
  To get the m0, m1, m2 microphones’s values.

9.8.1  Detailed Description

Accessing the microphone data.

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

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

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

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.8.2  Function Documentation

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

To get the m0, m1, m2 microphones’s values.
Parameters:

- \textit{m0} A pointer to store the m0 analogic value
- \textit{m1} A pointer to store the m1 analogic value
- \textit{m2} A pointer to store the m2 analogic value

\textbf{9.8.2.2} \texttt{void e_init_micro (void)}

Init the microphone A/D converter.

\textbf{Warning:}

- Must be called before starting using microphone
9.9  a_d/advance_ad_scan/e_micro.h File Reference

Accessing the microphone data.

Defines

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

Functions

- `int e_get_micro (unsigned int micro_id)`
  
  Get the last value of a given micro.

- `int e_get_micro_average (unsigned int micro_id, unsigned int filter_size)`
  
  Get the average on a given number of sample from a micro.

- `int e_get_micro_volume (unsigned int micro_id)`
  
  Get the difference between the highest and lowest sample.

9.9.1 Detailed Description

Accessing the microphone data.

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

Author:

Code: Borter Jean-Joel
Doc: Jonathan Besuchet

9.9.2 Define Documentation

9.9.2.1 `#define MIC0_BUFFER 0`

9.9.2.2 `#define MIC1_BUFFER 1`

9.9.2.3 `#define MIC2_BUFFER 2`

9.9.3 Function Documentation

9.9.3.1 `int e_get_micro (unsigned int micro_id)`

Get the last value of a given micro.
Parameters:

\texttt{micro\_id} micro’s ID (0, 1, or 2) (use \texttt{MIC0\_BUFFER} (p. 86), \texttt{MIC1\_BUFFER} (p. 86) , \texttt{MIC2\_BUFFER} (p. 86) defined in \texttt{e\_micro.h})

Returns:

result last value of the micro

\subsection{9.9.3.2 \texttt{int e\_get\_micro\_average} (unsigned int \texttt{micro\_id}, unsigned int \texttt{filter\_size})}

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

Parameters:

\texttt{micro\_id} micro’s ID (0, 1, or 2) (use \texttt{MIC0\_BUFFER} (p. 86), \texttt{MIC1\_BUFFER} (p. 86) , \texttt{MIC2\_BUFFER} (p. 86) defined in \texttt{e\_micro.h})

\texttt{filter\_size} number of sample to average

Returns:

result last value of the micro

\subsection{9.9.3.3 \texttt{int e\_get\_micro\_volume} (unsigned int \texttt{micro\_id})}

Get the difference between the highest and lowest sample.

Parameters:

\texttt{micro\_id} micro’s ID (0, 1, or 2) (use \texttt{MIC0\_BUFFER} (p. 86), \texttt{MIC1\_BUFFER} (p. 86) , \texttt{MIC2\_BUFFER} (p. 86) defined in \texttt{e\_micro.h})

Returns:

result volume
9.10  a_d/e_micro.h File Reference

Accessing the microphone data.

Functions

- void e_init_micro (void)
  Init the microphone A/D converter.

- void e_get_micro (int *m1, int *m2, int *m3)
  To get the m0, m1, m2 microphones’s values.

9.10.1  Detailed Description

Accessing the microphone data.

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

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

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

Author:

Code: Michael Bonani
    Doc: Jonathan Besuchet

9.10.2  Function Documentation

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

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

Parameters:

- m0  A pointer to store the m0 analogic value
9.10 a_d/e_micro.h File Reference

\begin{itemize}
\item \textit{m1} A pointer to store the m1 analogic value
\item \textit{m2} A pointer to store the m2 analogic value
\end{itemize}

9.10.2.2 \textbf{void e_init_micro (void)}

Init the microphone A/D converter.

\textbf{Warning:}

Must be called before starting using microphone
9.11 a_d/advance_ad_scan/e_prox.c File Reference

Accessing proximity sensor of e-puck.
#include "e_ad_conv.h"
#include "../../motor_led/e_epuck_ports.h"
#include "e_prox.h"

Functions

• void e_calibrate_ir ()
  
  To calibrate your ir sensor.

• int e_get_prox (unsigned int sensor_number)
  
  To get the analogic proxy sensor value of a specific ir sensor.

• int e_get_calibrated_prox (unsigned int sensor_number)
  
  To get the calibrated value of the ir sensor.

• int e_get_ambient_light (unsigned int sensor_number)
  
  To get the analogic ambient light value of a specific ir sensor.

Variables

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

9.11.1 Detailed Description

Accessing proximity sensor of e-puck.

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

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

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

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

Author:
Code: Lucas Meier
Doc: Jonathan Besuchet

9.11.2 Function Documentation

9.11.2.1 void e_calibrate_ir ()

To calibrate your ir sensor.

Warning:
Call this function one time before calling e_get_calibrated_prox

9.11.2.2 int e_get_ambient_light (unsigned int sensor_number)

To get the analogic ambient light value of a specific ir sensor.
This function return the analogic value of the ambient light measurement.

Parameters:

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

Returns:
The analogic value of the specified proxy sensor

9.11.2.3 int e_get_calibrated_prox (unsigned int sensor_number)

To get the calibrated value of the ir sensor.
This function return the calibrated analogic value of the ir sensor.

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

Parameters:

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

Returns:
The analogic value of the specified proxy sensor
9.11.2.4  int e_get_prox (unsigned int sensor_number)

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

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

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

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

Parameters:

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

Returns:

The analogic value of the specified proxy sensor

9.11.3  Variable Documentation

9.11.3.1  int e_ambient_and_reflected_ir[8]

Array to store the light when IR led is on

9.11.3.2  int e_ambient_ir[8]

Array to store the ambient light measurement

9.11.3.3  int init_value_ir[8] = {0,0,0,0,0,0,0,0}  [static]
9.12 a_d/e_prox.c File Reference

Accessing proximity sensor of e-puck with timer 1.

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

### Functions

- **void init_tmr1 (void)**
- **void _attribute_ ((interrupt, auto_psv, shadow))**
- **void e_init_prox (void)**
  
  *Init the proximity sensor A/D converter and the timer1.*

- **void e_stop_prox (void)**
  
  *Stop the acquisition (stop timer1).*

- **int e_get_prox (unsigned int sensor_number)**
  
  *To get the analogic proxy sensor value of a specific sensor.*

- **int e_get_ambient_light (unsigned int sensor_number)**
  
  *To get the analogic ambient light value of a specific sensor.*

### Variables

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

9.12.1 Detailed Description

Accessing proximity sensor of e-puck with timer 1.

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

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

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

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

Warning:
This module uses the timer1

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

9.12.2 Function Documentation

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

9.12.2.2 int e_get_ambient_light (unsigned int sensor_number)
To get the analogic ambient light value of a specific sensor.
To get the analogic ambient light value of a specific IR sensor.
This function returns the analogic value of the ambient light measurement.

Parameters:

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

Returns:
The analogic value of the specified proxy sensor

9.12.2.3 int e_get_prox (unsigned int sensor_number)
To get the analogic proxy sensor value of a specific sensor.
To get the analogic proxy sensor value of a specific IR sensor.
To estimate the proximity 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 obstacle is near.
9.12 a_d/e_prox.c File Reference

Parameters:

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

Returns:

The analogic value of the specified proxy sensor

9.12.2.4  void e_init_prox (void)

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

Warning:

Must be called before starting using proximity sensor

9.12.2.5  void e_stop_prox (void)

Stop the acquisition (stop timer1).

9.12.2.6  void init_tmr1 (void)

9.12.3  Variable Documentation

9.12.3.1  int ambient_and_reflected_ir[8]  [static]

9.12.3.2  int ambient_ir[8]  [static]

9.12.3.3  int reflected_ir[8]  [static]
9.13 a_d/advance_ad_scan/e_prox.h File Reference

Accessing proximity sensor of e-puck.

Functions

- **void** `e_calibrate_ir()`  
  To calibrate your ir sensor.

- **int** `e_get_prox(unsigned int sensor_number)`  
  To get the analogic proxy sensor value of a specific ir sensor.

- **int** `e_get_calibrated_prox(unsigned int sensor_number)`  
  To get the calibrated value of the ir sensor.

- **int** `e_get_ambient_light(unsigned int sensor_number)`  
  To get the analogic ambient light value of a specific ir sensor.

9.13.1 Detailed Description

Accessing proximity sensor of e-puck.

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

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

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

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

**Author:**

Code: Lucas Meier  
Doc: Jonathan Besuchet
9.13.2 Function Documentation

9.13.2.1 void e_calibrate_ir ()

To calibrate your ir sensor.

Warning:

Call this function one time before calling e_get_calibrated_prox

9.13.2.2 int e_get_ambient_light (unsigned int sensor_number)

To get the analogic ambient light value of a specific ir sensor.
This function return the analogic value of the ambient light measurement.

Parameters:

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

Returns:

The analogic value of the specified proxy sensor

To get the analogic ambient light value of a specific ir sensor.
This function retur the analogic value of the ambient light measurement.

Parameters:

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

Returns:

The analogic value of the specified proxy sensor

9.13.2.3 int e_get_calibrated_prox (unsigned int sensor_number)

To get the calibrated value of the ir sensor.
This function return the calbrated analogic value of the ir sensor.

Warning:

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

Parameters:

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

Returns:

The analogic value of the specified proxy sensor
To get the analogic proxy sensor value of a specific IR sensor.

To estimate the proximity 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 obstacle is near.

**Parameters:**

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

**Returns:**

The analogic value of the specified proxy sensor
9.14 a_d/e_prox.h File Reference

Accessing proximity sensor of e-puck with timer 1.

Functions

- void **e_init_prox** (void)
  
  *Init the proximity sensor A/D converter and the timer1.*

- void **e_stop_prox** (void)
  
  *Stop the acquisition (stop timer1).*

- int **e_get_prox** (unsigned int sensor_number)
  
  *To get the analogic proxy sensor value of a specific ir sensor.*

- int **e_get_ambient_light** (unsigned int sensor_number)
  
  *To get the analogic ambient light value of a specific ir sensor.*

9.14.1 Detailed Description

Accessing proximity sensor of e-puck with timer 1.

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

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

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

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

**Warning:**

This module uses the timer1

**Author:**

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

9.14.2.1 int e_get_ambient_light (unsigned int sensor_number)

To get the analogic ambient light value of a specific ir sensor.
This function return the analogic value of the ambient light measurement.

Parameters:

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

Returns:

The analogic value of the specified proxy sensor

9.14.2.2 int e_get_prox (unsigned int sensor_number)

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 obstacle is near.

Parameters:

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

Returns:

The analogic value of the specified proxy sensor
• 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 obstacle is near.

Parameters:

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

Returns:

  The analogic value of the specified proxy sensor

9.14.2.3 void e_init_prox (void)

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

Warning:

  Must be called before starting using proximity sensor

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

Warning:

  Must be called before starting using proximity sensor

9.14.2.4 void e_stop_prox (void)

Stop the acquisition (stop timer1).
9.15  a_d/e_accelerometer.c File Reference

Accessing the accelerometer sensor data.

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

Functions

- void `e_init_acc` (void)
  
  *Init the accelerometer A/D converter.*

- void `e_get_acc` (int *x, int *y, int *z)
  
  *To get the analogic x, y, z axis accelerations.*

9.15.1 Detailed Description

Accessing the accelerometer sensor data.

A little exemple to read the accelerator.

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

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

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet
9.15.2 Function Documentation

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

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

Parameters:

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

9.15.2.2 void e_init_acc (void)

Init the accelerometer A/D converter.

Warning:

  Must be called before starting using accelerometer
9.16  a_d/e_accelerometer.h File Reference

Accessing the accelerometer sensor data.

Functions

- void e_init_acc (void)
  
  Init the accelerometer A/D converter.
  
- void e_get_acc (int *x, int *y, int *z)
  
  To get the analogic x, y, z axis accelerations.

9.16.1  Detailed Description

Accessing the accelerometer sensor data.

A little exemple to read the accelerator.

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

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

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.16.2  Function Documentation

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

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

- \( x \): A pointer to store the analogic \( x \) acceleration
- \( y \): A pointer to store the analogic \( y \) acceleration
- \( z \): A pointer to store the analogic \( z \) acceleration

### 9.16.2.2 void e_init_acc (void)

Init the accelerometer A/D converter.

**Warning:**

Must be called before starting using accelerometer.
9.17 a_d/e_prox_timer2.c File Reference

Control proximity sensor of e-puck with timer2.

```c
#include "e_ad_conv.h"
#include ".../motor-led/e_epuck_ports.h"
#include "e_prox.h"
```

**Functions**

- void `init_tmr2` (void)
- void `__attribute__((interrupt, auto_psv, shadow))` `e_init_prox` (void)
  - Init the proximity sensor A/D converter and the timer2.
- void `e_stop_prox` (void)
  - Stop the acquisition (stop timer2).
- int `e_get_prox` (unsigned int sensor_number)
  - To get the analogic proxy sensor value of a specific sensor.
- int `e_get_ambient_light` (unsigned int sensor_number)
  - To get the analogic ambient light value of a specific sensor.

**Variables**

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

9.17.1 Detailed Description

Control proximity sensor of e-puck with timer2.

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

**Warning:**

The module uses the timer2

**See also:**

`e_prox.c`, `e_prox.h` to use the timer1

**Author:**

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

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

#### 9.17.2.2 `int e_get_ambient_light(unsigned int sensor_number)`

To get the analogic ambient light value of a specific sensor.
To get the analogic ambient light value of a specific ir sensor.
This function return the analogic value of the ambient light measurement.

**Parameters:**

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

**Returns:**

The analogic value of the specified proxy sensor

#### 9.17.2.3 `int e_get_prox(unsigned int sensor_number)`

To get the analogic proxy sensor value of a specific sensor.
To get the analogic proxy sensor value of a specific ir sensor.
To estimate the proxymity of an obstacle, we do the following things:

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

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

**Parameters:**

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

**Returns:**

The analogic value of the specified proxy sensor

#### 9.17.2.4 `void e_init_prox(void)`

Init the proxymity sensor A/D converter and the timer2.
Init the proxymity sensor A/D converter and the timer1.

**Warning:**

Must be called before starting using proximity sensor
9.17.2.5  void e_stop_prox (void)
Stop the acquisition (stop timer2).
Stop the acquisition (stop timer1).

9.17.2.6  void init_tmr2 (void)

9.17.3  Variable Documentation

9.17.3.1  int ambient_and_reflected_ir[8]  [static]
9.17.3.2  int ambient_ir[8]  [static]
9.17.3.3  int reflected_ir[8]  [static]
Manage Bluetooth.

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

Functions

• int e_bt_find_epuck (void)
  Try to find other e-puck.

• char e_bt_connect_epuck (void)

• int e_bt_reset (void)
  Reset the bluetooth module.

• char e_bt_factory_reset (void)
  Factory reset of the bluetooth module.

• char e_bt_transparent_mode (void)
  Change to transparent mode.

• void e_bt_exit_transparent_mode (void)
  Exit from the transparent mode.

• char e_bt_read_local_pin_number (char *PIN)
  Read the PIN number of this e-puck’s bluetooth module.

• char e_bt_read_local_name (char *name)
  Read the name of this e-puck’s bluetooth module.

• char e_bt_write_local_pin_number (char *PIN)
  Write the PIN number on this e-puck’s bluetooth module.

• char e_bt_write_local_name (char *name)
  Write the name on this e-puck’s bluetooth module.

• int e_bt_inquiry (struct BtDevice *device)
  Research all the accessible bluetooth devices.

• char e_bt_get_friendly_name (struct BtDevice *device)
  To get the friendly name of a bluetooth device.

• char e_bt_etablish_SPP_link (char *address)


Try to connect to another bluetooth device.

- char e_bt_release_SPP_link (void)
  
  Unconnect from the current bluetooth device.

- char e_bt_send_SPP_data (char *data, char datalength)
  
  Send data to the current bluetooth device.

- char e_bt_list_local_paired_device (void)
  
  Make a list of the bluetooth address of paired device.

- char e_bt_remove_local_paired_device (int j)
  
  Remove a paired bluetooth device.

Variables

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

9.18.1 Detailed Description

Manage Bluetooth.

This module manage the Bluetooth device.

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.18.2 Function Documentation

9.18.2.1 char e_bt_connect_epuck (void)

9.18.2.2 char e_bt_establish_SPP_link (char * address)

Try to connect to another bluetooth device.

Parameters:

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

Returns:

  bluetooth error if one occur, 0 otherwise

9.18.2.3 void e_bt_exit_transparent_mode (void)

Exit from the transparent mode.
### 9.18.2.4 char e_bt_factory_reset (void)

Factory reset of the bluetooth module.

**Warning:**

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

**Returns:**

bluetooth error if one occur, 0 otherwise

### 9.18.2.5 int e_bt_find_epuck (void)

Try to find other e-puck.

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

**Returns:**

number of e-puck found

See also:

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

### 9.18.2.6 char e_bt_get_friendly_name (struct BtDevice *device)

To get the friendly name of a bluetooth device.

**Parameters:**

*device* A pointer on the device that you want the name

**Returns:**

bluetooth error if one occur, 0 otherwise

See also:

BtDevice (p. 59)

### 9.18.2.7 int e_bt_inquiry (struct BtDevice *device)

Research all the accessible bluetooth devices.

**Parameters:**

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

**Returns:**

the number of device found

See also:

BtDevice (p. 59), e_bt_present_device (p. 120)
9.18.2.8  char e_bt_list_local_paired_device (void)
Make a list of the bluetooth address of paired device.

Returns:
The number of device found

See also:
e_bt_local_paired_device (p. 120)

9.18.2.9  char e_bt_read_local_name (char * name)
Read the name of this e-puck’s bluetooth module.

Parameters:
name  A pointer to store the name

Returns:
bluetooth error if one occur, 0 otherwise

9.18.2.10 char e_bt_read_local_pin_number (char * PIN)
Read the PIN number of this e-puck’s bluetooth module.

Parameters:
PIN  A pointer to store the PIN number

Returns:
bluetooth error if one occur, 0 otherwise

9.18.2.11 char e_bt_release_SPP_link (void)
Unconnect from the current bluetooth device.

Returns:
bluetooth error if one occur, 0 otherwise

9.18.2.12 char e_bt_remove_local_paired_device (int j)
Remove a paired bluetooth device.

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

Returns:
bluetooth error if one occur, 0 otherwise
9.18.2.13  int e_bt_reset (void)

Reset the bluetooth module.

Returns:

The version number

9.18.2.14  char e_bt_send_SPP_data (char * data, char datalength)

Send data to the current bluetooth device.

Warning:

send maximum 127 bytes if you are in non transparent mode

Parameters:

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

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.15  char e_bt_transparent_mode (void)

Change to transparent mode.

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.16  char e_bt_write_local_name (char * name)

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

Parameters:

- name  A pointer to store the name

Returns:

bluetooth error if one occur, 0 otherwise

9.18.2.17  char e_bt_write_local_pin_number (char * PIN)

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

Parameters:

- PIN  A pointer to store the PIN number

Returns:

bluetooth error if one occur, 0 otherwise
9.18.3 Variable Documentation

9.18.3.1 unsigned char e_bt_local_paired_device[6 *8]

9.18.3.2 struct BtDevice e_bt_present_device[10]

An extern array containing all the bluetooth device detected. It’s carried out by the fonction e_bt_find_epuck

See also:

  e_bt_find_epuck (p. 117)

9.18.3.3 struct BtEPuck e_bt_present_epuck[10]

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

See also:

  e_bt_find_epuck (p. 117)

9.18.3.4 char local_bt_PIN[4]
Data Structures

- struct BtDevice
  
  general struct for bluetooth device

- struct BtEPuck
  
  general struct for other e-puck

Functions

- int e_bt_find_epuck (void)
  
  Try to find other e-puck.

- char e_bt_connect_epuck (void)
- char e_bt_read_local_pin_number (char *PIN)
  
  Read the PIN number of this e-puck’s bluetooth module.

- char e_bt_read_local_name (char *name)
  
  Read the name of this e-puck’s bluetooth module.

- char e_bt_write_local_pin_number (char *PIN)
  
  Write the PIN number on this e-puck’s bluetooth module.

- char e_bt_write_local_name (char *name)
  
  Write the name on this e-puck’s bluetooth module.

- char e_bt_etablish_SPP_link (char *address)
  
  Try to connect to another bluetooth device.

- char e_bt_release_SPP_link (void)
  
  Unconnect from the current bluetooth device.

- char e_bt_send_SPP_data (char *data, char datalenght)
  
  Send data to the current bluetooth device.

- char e_bt_transparent_mode (void)
  
  Change to transparent mode.

- void e_bt_exit_transparent_mode (void)
  
  Exit from the transparent mode.

- char e_bt_list_local_paired_device (void)
  
  Make a list of the bluetooth address of paired device.
• **char e_bt_remove_local_paired_device (int)**
  
  Remove a paired bluetooth device.

• **int e_bt_inquiry (struct BtDevice *device)**
  
  Research all the accessible bluetooth devices.

• **char e_bt_get_friendly_name (struct BtDevice *device)**
  
  To get the friendly name of a bluetooth device.

• **int e_bt_reset (void)**
  
  Reset the bluetooth module.

• **char e_bt_factory_reset (void)**
  
  Factory reset of the bluetooth module.

### Variables

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

#### 9.19.1 Detailed Description

Manage Bluetooth.

This module manage the bluetooth device.

**Author:**

Code: Michael Bonani
Doc: Jonathan Besuchet

#### 9.19.2 Function Documentation

##### 9.19.2.1 char e_bt_connect_epuck (void)

##### 9.19.2.2 char e_bt_etablish_SPP_link (char * address)

Try to connect to another bluetooth device.

**Parameters:**

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

**Returns:**

  - bluetooth error if one occur, 0 otherwise

##### 9.19.2.3 void e_bt_exit_transparent_mode (void)

Exit from the transparent mode.
9.19.2.4 char e_bt_factory_reset (void)
Factory reset of the bluetooth module.

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

Returns:
bluetooth error if one occur, 0 otherwise

9.19.2.5 int e_bt_find_epuck (void)
Try to find other e-puck.
This function make global inquiry and check which device are e-puck, and list them in globales tables.

Returns:
number of e-puck found

See also:
e_bt_present_device (p. 120), e_bt_present_epuck (p. 120)

9.19.2.6 char e_bt_get_friendly_name (struct BtDevice * device)
To get the friendly name of a bluetooth device.

Parameters:

device A pointer on the device that you want the name

Returns:
bluetooth error if one occur, 0 otherwise

See also:
BtDevice (p. 59)

9.19.2.7 int e_bt_inquiry (struct BtDevice * device)
Research all the accessible bluetooth devices.

Parameters:

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

Returns:
the number of device found

See also:
BtDevice (p. 59), e_bt_present_device (p. 120)
9.19.2.8  char e_bt_list_local_paired_device (void)

Make a list of the bluetooth address of paired device.

Returns:
   The number of device found

See also:
   e_bt_local_paired_device (p. 120)

9.19.2.9  char e_bt_read_local_name (char * name)

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

Parameters:
   name   A pointer to store the name

Returns:
   bluetooth error if one occur, 0 otherwise

9.19.2.10  char e_bt_read_local_pin_number (char * PIN)

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

Parameters:
   PIN   A pointer to store the PIN number

Returns:
   bluetooth error if one occur, 0 otherwise

9.19.2.11  char e_bt_release_SPP_link (void)

Unconnect from the current bluetooth device.

Returns:
   bluetooth error if one occur, 0 otherwise

9.19.2.12  char e_bt_remove_local_paired_device (int j)

Remove a paired bluetooth device.

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

Returns:
   bluetooth error if one occur, 0 otherwise
9.19.2.13 int e_bt_reset (void)

Reset the bluetooth module.

Returns:

The version number

9.19.2.14 char e_bt_send_SPP_data (char *data, char datalength)

Send data to the current bluetooth device.

Warning:

send maximum 127 bytes if you are in non transparent mode

Parameters:

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

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.15 char e_bt_tranparent_mode (void)

Change to transparent mode.

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.16 char e_bt_write_local_name (char *name)

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

Parameters:

name  A pointer to store the name

Returns:

bluetooth error if one occur, 0 otherwise

9.19.2.17 char e_bt_write_local_pin_number (char *PIN)

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

Parameters:

PIN  A pointer to store the PIN number

Returns:

bluetooth error if one occur, 0 otherwise
9.19.3 Variable Documentation

9.19.3.1 unsigned char e_bt_local_paired_device[6 +8]

9.19.3.2 struct BtDevice e_bt_present_device[10]

An extern array containing all the bluetooth device detected. It’s carried out by the fonction e_bt_find_epuck

See also:
   e_bt_find_epuck (p. 117)

9.19.3.3 struct BtEPuck e_bt_present_epuck[10]

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

See also:
   e_bt_find_epuck (p. 117)
Calculate the timing for the camera (two timers).

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

**Defines**

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

**Functions**

- `int e_po3030k_config_cam (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)`
- `int e_po3030k_get_bytes_per_pixel (int color_mode)`
- `void e_po3030k_init_cam (void)`

### 9.20.1 Detailed Description

Calculate the timing for the camera (two timers).

**Author:**

Philippe Retornaz

### 9.20.2 Define Documentation

9.20.2.1 `#define ARRAY_ORIGINE_X 210`

9.20.2.2 `#define ARRAY_ORIGINE_Y 7`

9.20.2.3 `#define IRQ_PIX_LAT 1`

### 9.20.3 Function Documentation

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

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

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

Parameters:

- `sensor_x1` The X coordinate of the window’s corner
- `sensor_y1` The Y coordinate of the window’s corner
- `sensor_width` The Width of the interest area, in FULL sampling scale
- `sensor_height` The Height of the interest area, in FULL sampling scale
- `zoom_fact_width` The subsampling to apply for the window’s Width
- `zoom_fact_height` The subsampling to apply for the window’s Height
- `color_mode` The color mode in which the camera should be configured

Returns:

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

See also:

- e_po3030k_write_cam_registers (p. 187)

9.20.3.2 int e_po3030k_get_bytes_per_pixel (int color_mode)

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

Parameters:

- `color_mode` The given color mode

Returns:

The number of bytes per pixel in the given color mode

9.20.3.3 void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function
Calculate the timing for the camera (three timers).

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

**Defines**

- `#define ARRAY_ORIGINE_X 210`
- `#define ARRAY_ORIGINE_Y 7`

**Functions**

- `int e_po3030k_config_cam` (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)
- `int e_po3030k_get_bytes_per_pixel` (int color_mode)
- `void e_po3030k_init_cam` (void)

### 9.21.1 Detailed Description

Calculate the timing for the camera (three timers).

### 9.21.2 Define Documentation

#### 9.21.2.1 `#define ARRAY_ORIGINE_X 210`

#### 9.21.2.2 `#define ARRAY_ORIGINE_Y 7`

### 9.21.3 Function Documentation

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

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

**Warning:**

If the common denominator 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 respectively 4 or 2.

**Parameters:**

- `sensor_x1` The X coordinate of the window’s corner
- `sensor_y1` The Y coordinate of the window’s corner
sensor_width  the Width of the interest area, in FULL sampling scale
sensor_height  The Height of the interest area, in FULL sampling scale
zoom_fact_width  The subsampling to apply for the window’s Width
zoom_fact_height  The subsampling to apply for the window’s Height
color_mode  The color mode in which the camera should be configured

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

See also:
e_po3030k_write_cam_registers (p. 187)

9.21.3.2  int e_po3030k_get_bytes_per_pixel (int color_mode)
Return the number of bytes per pixel in the given color mode

Parameters:
color_mode  The given color mode

Returns:
The number of bytes per pixel in the given color mode

9.21.3.3  void e_po3030k_init_cam (void)
Initialize the camera, must be called before any other function
Defines

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

Functions

- int e_po3030k_config_cam (unsigned int sensor_x1, unsigned int sensor_y1, unsigned int sensor_width, unsigned int sensor_height, unsigned int zoom_fact_width, unsigned int zoom_fact_height, int color_mode)
- int e_po3030k_get_bytes_per_pixel (int color_mode)
- void e_po3030k_init_cam (void)
- void e_po3030k_write_cam_registers (void)
- void e_po3030k_launch_capture (char *buf)
- int e_po3030k_apply_timer_config (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)
- int e_po3030k_is_img_ready (void)
- int e_po3030k_set_color_mode (int mode)
- int e_po3030k_set_sampling_mode (int mode)
- int e_po3030k_set_speed (int mode)
- int e_po3030k_set_wx (unsigned int start, unsigned int stop)
- int e_po3030k_set_wy (unsigned int start, unsigned int stop)
- int e_po3030k_set_vsync (unsigned int start, unsigned int stop, unsigned int col)
- void e_po3030k_read_cam_registers (void)
- int e_po3030k_set_register (unsigned char adr, unsigned char value)
- int e_po3030k_get_register (unsigned char adr, unsigned char *value)
- void e_po3030k_set_bias (unsigned char pixbias, unsigned char opbias)
- void e_po3030k_set_integr_time (unsigned long time)
- void e_po3030k_set_mirror (int vertical, int horizontal)
- void e_po3030k_set_adc_offset (unsigned char offset)
- void e_po3030k_set_sepia (int status)
• void e_po3030k_set_lens_gain (unsigned char red, unsigned char green, unsigned char blue)
• void e_po3030k_set_edge_prop (unsigned char gain, unsigned char tresh)
• void e_po3030k_set_gamma_coef (unsigned char array[12], char color)
• void e_po3030k_write_gamma_coef (void)
• int e_po3030k_sync_register_array (unsigned char start, unsigned char stop)
• void e_po3030k_set_color_matrix (unsigned char array[3 * 3])
• void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)
• void e_po3030k_set_brigh_contr (unsigned char bright, unsigned char contrast)
• void e_po3030k_set_sepia_tone (unsigned char cb, unsigned char cr)
• void e_po3030k_set_ww (unsigned char ww)
• void e_po3030k_set_awb_ae_tol (unsigned char awbm, unsigned char aem)
• void e_po3030k_set_ae_speed (unsigned char b, unsigned char d)
• void e_po3030k_set_exposure (long t)
• void e_po3030k_set_ref_exposure (unsigned char exp)
• void e_po3030k_set_max_min_exp (unsigned int min, unsigned int max)
• void e_po3030k_set_max_min_awb (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratiob, unsigned char ratiob)
• int e_po3030k_set_weight_win (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
• void e_po3030k_set_awb_ae (int awb, int ae)
• int e_po3030k_set_color_gain (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
• void e_po3030k_set_flicker_mode (int manual)
• void e_po3030k_set_flicker_detection (int hz50, int hz60)
• int e_po3030k_set_flicker_man_set (int hz50, int hz60, int fdm, int fk, int tol)

9.22.1 Detailed Description

PO3030k library header (two timers).

Author:
Philippe Retornaz

9.22.2 Define Documentation

9.22.2.1 #define ARRAY_HEIGHT 480
9.22.2.2 #define ARRAY_WIDTH 640
9.22.2.3 #define GREY_SCALE_MODE 0
9.22.2.4 #define MODE_QQVGA 0x33
9.22.2.5 #define MODE_QVGA 0x11
9.22.2.6 #define MODE_VGA 0x44
9.22.2.7 #define PO3030K_FULL 1

If you set this at 0, you save about 168 bytes of memory But you loose all advanced camera functions
9.22.8  #define RGB_565_MODE 1
9.22.9  #define SPEED_128 0x70
9.22.10 #define SPEED_16 0x40
9.22.11 #define SPEED_2 0x00
9.22.12 #define SPEED_2_3 0x10
9.22.13 #define SPEED_32 0x50
9.22.14 #define SPEED_4 0x20
9.22.15 #define SPEED_64 0x60
9.22.16 #define SPEED_8 0x30
9.22.17 #define YUV_MODE 2

9.22.3  Function Documentation

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

Modify the interrupt configuration

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

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

Returns:

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

See also:

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

Modify the interrupt configuration

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

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

See also:

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

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

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

Warning:

If the common denominator 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 respectively 4 or 2. Moreover greyscale is twice faster than color mode.

Parameters:

sensor_x1  The X coordinate of the window’s corner
sensor_y1  The Y coordinate of the window’s corner
sensor_width  the Width of the interest area, in FULL sampling scale
sensor_height  The Height of the interest area, in FULL sampling scale
zoom_fact_width  The subsampling to apply for the window’s Width
zoom_fact_height  The subsampling to apply for the window’s Height
color_mode  The color mode in which the camera should be configured

Returns:

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

See also:

e_po3030k_write_cam_registers (p. 187)

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

Warning:

If the common denominator 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 respectively 4 or 2.

Parameters:

sensor_x1  The X coordinate of the window’s corner
sensor_y1  The Y coordinate of the window’s corner
sensor_width  the Width of the interest area, in FULL sampling scale
sensor height The Height of the interest area, in FULL sampling scale

zoom fact width The subsampling to apply for the window’s Width

zoom fact height The subsampling to apply for the window’s Height

color_mode The color mode in which the camera should be configured

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

See also:
e_po3030k_write_cam_registers (p. 187)

9.22.3.3 int e_po3030k_get_bytes_per_pixel (int color_mode)

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

Parameters:

color_mode The given color mode

Returns:
The number of bytes per pixel in the given color mode

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

Get the register adr value

Parameters:

adr The address

value The pointer to the value to write to

Returns:
Zero if register found, non-zero if not found

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

See also:
e_po3030k_set_register (p. 183)

9.22.3.5 void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function
9.22.3.6  int e_po3030k_is_img_ready (void)

Check if the current capture is finished

Returns:

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

See also:

   e_po3030k_launch_capture (p. 193)

9.22.3.7  void e_po3030k_launch_capture (char *buf)

Launch a capture in the buf buffer

Parameters:

   buf  The buffer to write to

See also:

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

9.22.3.8  void e_po3030k_read_cam_registers (void)

Read the camera register

See also:

   e_po3030k_write_cam_registers (p. 187)

9.22.3.9  void e_po3030k_set_adc_offset (unsigned char offset)

Set the Analog to Digital Converter offset

Parameters:

   offset  The offset

See also:

   Datasheet p.28

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

Set AE speed

Parameters:

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

See also:

   Datasheet p.44
9.22.3.11  void e_po3030k_set_awb_ae (int awb, int ae)

Enable/Disable AWB and AE

Parameters:
  
a wb  1 mean AWB enabled, 0 mean disabled
  ae  1 mean AE enabled, 0 mean disabled

See also:
  Datasheet p. 50

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

Set AWB/AE tolerence margin

Parameters:
  
a wbm  AWV Margin ( 4 lower bits only )
  aem  AW Margin ( 4 lower bits only )

See also:
  Datasheet p.44

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

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

Parameters:
  
pixbias  The pixel bias
  opbias  The Amplificator bias

See also:
  Datasheet p.22

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

Set the Brightness & Contrast

Parameters:
  
b right  The Brightness ( signed 1+7bits fixed point format )
  contrast  The Contrast

See also:
  Datasheet p. 40
9.22.3.15  void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)
Set the color gain (Cb/Cr)

Parameters:
  cg11c  Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])
  cg22c  Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:
  Datasheet p. 40

9.22.3.16  int e_po3030k_set_color_gain (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
Set the gains of the camera

Parameters:
  global  The global gain ∈ {0, 79}
  red    The red pixel’s gain (fixed point [2:6] format)
  green1 The green pixel near read one gain ([2:6] format)
  green2 The green pixel near blue one gain ([2:6] format)
  blue   The blue pixel’s gain ([2:6] format)

Returns:
  Zero if OK, non-zero if an error occur

See also:
  Datasheet p.23-24

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

9.22.3.18  int e_po3030k_set_color_mode (int mode)
Set the camera color mode

Warning:
  This is an internal function, use e_po3030k_config_cam

Parameters:
  mode  The color mode

Returns:
  Zero if OK, non-zero if an error occur

See also:
  Datasheet p. 31, e_po3030k_write_cam_registers (p. 187) and e_po3030k_config_cam (p. 144)
9.22 camera/fast_2_timer/e_po3030k.h File Reference

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

Set Edge properties

Parameters:

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

See also:

  Datasheet p.36

9.22.3.20  void e_po3030k_set_exposure (long t)

Set exposure time

Parameters:

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

Warning:

  Only writable if AE is disabled

See also:

  Datasheet p.45

9.22.3.21  void e_po3030k_set_flicker_detection (int hz50, int hz60)

Set the 50/60Hz flicker detection

Parameters:

  hz50  Non-zero mean 50Hz flicker detection enabled (default disabled)
  hz60  Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

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

See also:

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

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

Set the camera’s manual flicker’s detection setting

Parameters:

  hz50  The Hz for the 50Hz detection
hz60  The Hz for the 60Hz detection
fdm  Flicker duration mode
fk  Flicker count step
tol  Flicker tolerance

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

Returns:
Non-zero if an error occur, 0 if OK

See also:
_datasheet_p.29-30  _e_po3030k_set_flicker_detection()_  (p. 133)  _e_po3030k_set_flicker_mode()_  (p. 134)

9.22.3.23  _void e_po3030k_set_flicker_mode (int manual)_
Set flicker detection mode

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

See also:
_datasheet_p.29  _e_po3030k_set_flicker_detection()_  (p. 133)  _e_po3030k_set_flicker_man_set()_  (p. 133)

9.22.3.24  _void e_po3030k_set_gamma_coef (unsigned char array[12], char color)_
Set gamma coefficient

Warning:
This feature need extra care from the user

Parameters:
_array  Gamma coefficient array
_color  First two bytes : - 0b01 => Green
   * 0b00 => Red
   * else => Blue

See also:
_datasheet_p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef
9.22.3.25  void e_po3030k_set_integr_time (unsigned long time)

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

Parameters:

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

See also:

  Datasheet p.25

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

Set lens shading gain.

Parameters:

  red  Lens gain for red pixel ∈ {0, 15}
  green  Lens gain for green pixel ∈ {0, 15}
  blue  Lens gain for blue pixel ∈ {0, 15}

See also:

  Datasheet p.36

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

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

Parameters:

  minb  The minimum blue gain
  maxb  The maximum blue gain
  minr  The minimum red gain
  maxr  The maximum red gain
  ratiob  The red gain ratio
  ratiob  The blue gain ratio

See also:

  Datasheet p. 47-48
9.22.3.28  void e_po3030k_set_max_min_exp (unsigned int max, unsigned int min)

Set the minimum and maximum exposure time in AE mode

Parameters:

  * min The minimum exposure time
  * max The maximum exposure time

See also:

  * Datasheet p.46-47

9.22.3.29  void e_po3030k_set_mirror (int vertical, int horizontal)

Enable/Disable horizontal or vertical mirror

Parameters:

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

See also:

  * Datasheet p.27

9.22.3.30  void e_po3030k_set_ref_exposure (unsigned char exp)

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

Parameters:

  * exp The target exposure level

See also:

  * Datasheet p.45

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

Set the register adr to value value

Parameters:

  * adr The address
  * value The value

Returns:

  * Zero if register found, non-zero if not found

Warning:

  * This function is sub-optimal, if you use it heavily add an internal function to register.c
9.22.3.32  int e_po3030k_set_sampling_mode (int mode)

Set the camera sampling mode

Warning:
This is an internal function, use e_po3030k_config_cam

Parameters:
mode  The given sampling mode

Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p. 28 and e_po3030k_config_cam (p. 144)

9.22.3.33  void e_po3030k_set_sepia (int status)

Enable/Disable Sepia color

Parameters:
status  Set status to 1 to enable, 0 to disable

See also:
Datasheet p.34 and 74

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

Set The color tone at sepia color condition

Parameters:
cb  Cb tone
cr  Cr tone

See also:
e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74
9.22.3.35  int e_po3030k_set_speed (int mode)

Set the camera speed

Warning:
   This is an internal function, use e_po3030k_config_cam

Parameters:
   mode   The given speed

Returns:
   Zero if OK, non-zero if unknown mode

See also:
   Datasheet p. 26 and e_po3030k_config_cam (p. 144)

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

Set the camera window VSYNC coordinate

Warning:
   This is an internal function, use e_po3030k_ConfigCam

Parameters:
   start  The start row
   stop   The stop row
   col    The start/stop column

Returns:
   Zero if OK, non-zero if an error occur

See also:
   Datasheet p.42-43, e_po3030k_write_cam_registers (p. 187) and e_po3030k_config_cam (p. 144)

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

Set the Weighting Window coordinate

Parameters:
   x1    The X1 coordinate \( \in \{211, x2\} \)
   x2    The X2 coordinate \( \in \{x1 + 1, 423\} \)
   y1    The Y1 coordinate \( \in \{160, y2\} \)
   y2    The Y2 coordinate \( \in \{y1 + 1, 319\} \)
Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 49

9.22.3.38 void e_po3030k_set_ww (unsigned char ww)

Set the Center weight (Back Light compensation) Control parameter

Parameters:

ww  Center weight ( 4 lower bits only )

See also:

Datasheet p.44

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

Set the camera window X coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start  The start column
stop   The stop column

Returns:

Zero if OK, non-zero if an error occur

See also:

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

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

Set the camera window Y coordinate

Warning:

This is an internal function, use e_po3030k_ConfigCam

Parameters:

start  The start row
stop   The stop row
9.22.3.41 int e_po3030k_sync_register_array (unsigned char start, unsigned char stop)

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

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

Parameters:

start The beginning address of the write
stop The last write address

Returns:
The number of register written

See also:

e_po3030k_write_cam_registers (p. 187)

9.22.3.42 void e_po3030k_write_cam_registers (void)

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

See also:

e_po3030k_read_cam_registers (p. 177)

9.22.3.43 void e_po3030k_write_gamma_coef (void)

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

Warning:
This function need extra care from the user

See also:

e_po3030k_set_gamma_coef (p. 181)
PO3030k library header (three timers).

Defines

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

Functions

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

PO3030k library header (three timers).

Author:
Philippe Retornaz

9.23.2 Define Documentation

9.23.2.1 #define ARRAY_HEIGHT 480

9.23.2.2 #define ARRAY_WIDTH 640

9.23.2.3 #define GREY_SCALE_MODE 0

9.23.2.4 #define MODE_QQVGA 0x33

9.23.2.5 #define MODE_QVGA 0x11

9.23.2.6 #define MODE_VGA 0x44

9.23.2.7 #define PO3030K_FULL 1

If you set this at 0, you save about 168 bytes of memory. But you loose all advanced camera functions.
9.23.2.8 #define RGB_565_MODE 1
9.23.2.9 #define SPEED_128 0x70
9.23.2.10 #define SPEED_16 0x40
9.23.2.11 #define SPEED_2 0x00
9.23.2.12 #define SPEED_2_3 0x10
9.23.2.13 #define SPEED_32 0x50
9.23.2.14 #define SPEED_4 0x20
9.23.2.15 #define SPEED_64 0x60
9.23.2.16 #define SPEED_8 0x30
9.23.2.17 #define YUV_MODE 2

9.23.3 Function Documentation

9.23.3.1 void e_po3030k_apply_timer_config (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)
Modify the interrupt configuration

Warning:
This is an internal function, use e_po3030k_config_cam

Parameters:

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

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

See also:
e_po3030k_get_bytes_per_pixel (p. 145) and e_po3030k_config_cam (p. 144)

Modify the interrupt configuration

Warning:
This is an internal function, use e_po3030k_config_cam

Parameters:

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

See also:
e_po3030k_get_bytes_per_pixel (p. 145) and e_po3030k_config_cam (p. 144)

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

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

Warning:

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

Parameters:

  sensor_x1  The X coordinate of the window’s corner
  sensor_y1  The Y coordinate of the window’s corner
  sensor_width the Width of the interest area, in FULL sampling scale
  sensor_height The Height of the interest area, in FULL sampling scale
  zoom_fact_width The subsampling to apply for the window’s Width
  zoom_fact_height The subsampling to apply for the window’s Height
  color_mode The color mode in which the camera should be configured

Returns:

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

See also:
e_po3030k_write_cam_registers (p. 187)

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

Warning:

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

Parameters:

  sensor_x1  The X coordinate of the window’s corner
  sensor_y1  The Y coordinate of the window’s corner
  sensor_width the Width of the interest area, in FULL sampling scale
sensor_height  The Height of the interest area, in FULL sampling scale

zoom_fact_width  The subsampling to apply for the window’s Width

zoom_fact_height The subsampling to apply for the window’s Height

color_mode  The color mode in which the camera should be configured

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

See also:
e_po3030k_write_cam_registers (p. 187)

9.23.3.3  int e_po3030k_get_bytes_per_pixel (int color_mode)

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

Parameters:

color_mode  The given color mode

Returns:
The number of bytes per pixel in the given color mode

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

Get the register adr value

Parameters:

adr  The address

value  The pointer to the value to write to

Returns:
Zero if register found, non-zero if not found

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

See also:
e_po3030k_set_register (p. 183)

9.23.3.5  void e_po3030k_init_cam (void)

Initialize the camera, must be called before any other function
9.23.3.6 int e_po3030k_is_img_ready (void)

Check if the current capture is finished

**Returns:**

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

**See also:**

- `e_po3030k_launch_capture` (p. 193)

9.23.3.7 void e_po3030k_launch_capture (char ∗buf)

Launch a capture in the `buf` buffer

**Parameters:**

- `buf` The buffer to write to

**See also:**

- `e_po3030k_config_cam` (p. 144) and `e_po3030k_is_img_ready` (p. 193)

9.23.3.8 void e_po3030k_read_cam_registers (void)

Read the camera register

**See also:**

- `e_po3030k_write_cam_registers` (p. 187)

9.23.3.9 void e_po3030k_set_adc_offset (unsigned char offset)

Set the Analog to Digital Converter offset

**Parameters:**

- `offset` The offset

**See also:**

Datasheet p.28

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

Set AE speed

**Parameters:**

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

**See also:**

Datasheet p.44
9.23.3.11  void e_po3030k_set_awb_ae (int awb, int ae)

Enable/Disable AWB and AE

Parameters:

awb  1 mean AWB enabled, 0 mean disabled
ae   1 mean AE enabled, 0 mean disabled

See also:

Datasheet p. 50

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

Set AWB/AE tolerance margin

Parameters:

awbm AWV Margin (4 lower bits only)
aem  AW Margin (4 lower bits only)

See also:

Datasheet p.44

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

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

Parameters:

pixbias  The pixel bias
opbias  The Amplificator bias

See also:

Datasheet p.22

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

Set the Brightness & Contrast

Parameters:

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

See also:

Datasheet p. 40
9.23.3.15  void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)

Set The color gain (Cb/Cr)

Parameters:

cg11c  Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])
cg22c  Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:

Datasheet p. 40

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

Set the gains of the camera

Parameters:

global  The global gain ∈ {0, 79}
red     The red pixel’s gain (fixed point [2:6] format)
green1  The green pixel near read one gain ([2:6] format)
green2  The green pixel near blue one gain ([2:6] format)
blue    The blue pixel’s gain ([2:6] format)

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p.23-24

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

9.23.3.18  int e_po3030k_set_color_mode (int mode)

Set the camera color mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode   The color mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 31, e_po3030k_write_cam_registers (p. 187) and e_po3030k_config_cam (p. 144)
9.23.3.19 void e_po3030k_set_edge_prop (unsigned char gain, unsigned char tresh)

Set Edge properties

Parameters:

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

See also:

    Datasheet p.36

9.23.3.20 void e_po3030k_set_exposure (long t)

Set exposure time

Parameters:

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

Warning:

    Only writable if AE is disabled

See also:

    Datasheet p.45

9.23.3.21 void e_po3030k_set_flicker_detection (int hz50, int hz60)

Set the 50/60Hz flicker detection

Parameters:

    hz50  Non-zero mean 50Hz flicker detection enabled (default disabled)
    hz60  Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

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

See also:

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

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

Set the camera’s manual flicker’s detection setting

Parameters:

    hz50  The Hz for the 50Hz detection
hz60  The Hz for the 60Hz detection
fdm  Flicker duration mode
fk  Flicker count step
tol  Flicker tolerance

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

Returns:
Non-zero if an error occur, 0 if OK

See also:
Datasheet p.29-30  e_po3030k_set_flicker_detection() (p. 133) e_po3030k_set_flicker_mode()
(p. 134)

9.23.3.23  void e_po3030k_set_flicker_mode (int manual)
Set flicker detection mode

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

See also:
Datasheet p.29  e_po3030k_set_flicker_detection() (p. 133) e_po3030k_set_flicker_man_set()
(p. 134)

9.23.3.24  void e_po3030k_set_gamma_coef (unsigned char array[12], char color)
Set gamma coefficient

Warning:
This feature need extra care from the user

Parameters:
array  Gamma coefficient array
color  First two bytes: 0b01 => Green
      0b00 => Red
      else => Blue

See also:
Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef
9.23.3.25 `void e_po3030k_set_integr_time (unsigned long time)`

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

**Parameters:**
- `time` The integration time (fixed point [14:6] format)

**See also:**
- Datasheet p.25

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

Set lens shading gain.

**Parameters:**
- `red` Lens gain for red pixel $\in \{0, 15\}$
- `green` Lens gain for green pixel $\in \{0, 15\}$
- `blue` Lens gain for blue pixel $\in \{0, 15\}$

**See also:**
- Datasheet p.36

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

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

**Parameters:**
- `minb` The minimum blue gain
- `maxb` The maximum blue gain
- `minr` The minimum red gain
- `maxr` The maximum red gain
- `ratior` The red gain ratio
- `ratiob` The blue gain ratio

**See also:**
- Datasheet p. 47-48
9.23.3.28  void e_po3030k_set_max_min_exp (unsigned int max, unsigned int min)
Set the minimum and maximum exposure time in AE mode

Parameters:

  min  The minimum exposure time
  max  The maximum exposure time

See also:

  Datasheet p.46-47

9.23.3.29  void e_po3030k_set_mirror (int vertical, int horizontal)
Enable/Disable horizontal or vertical mirror

Parameters:

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

See also:

  Datasheet p.27

9.23.3.30  void e_po3030k_set_ref_exposure (unsigned char exp)
Set the reference exposure. The average brightness which the AE should have

Parameters:

  exp  The target exposure level

See also:

  Datasheet p.45

9.23.3.31  int e_po3030k_set_register (unsigned char adr, unsigned char value)
Set the register adr to value value

Parameters:

  adr  The address
  value  The value

Returns:

  Zero if register found, non-zero if not found

Warning:

  This function is sub-optimal, if you use it heavily add an internal function to register.c
9.23.3.32 int e_po3030k_set_sampling_mode (int mode)
Set the camera sampling mode

Warning:
This is an internal function, use e_po3030k_config_cam

Parameters:
mode The given sampling mode

Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p. 28 and e_po3030k_config_cam (p. 144)

9.23.3.33 void e_po3030k_set_sepia (int status)
Enable/Disable Sepia color

Parameters:
status Set status to 1 to enable, 0 to disable

See also:
Datasheet p.34 and 74

9.23.3.34 void e_po3030k_set_sepia_tone (unsigned char cb, unsigned char cr)
Set The color tone at sepia color condition

Parameters:
  cb Cb tone
  cr Cr tone

See also:
e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74
9.23.3.35 int e_po3030k_set_speed (int mode)
Set the camera speed

**Warning:**
This is an internal function, use e_po3030k_config_cam

**Parameters:**
mode The given speed

**Returns:**
Zero if OK, non-zero if unknow mode

**See also:**
Datasheet p. 26 and e_po3030k_config_cam (p. 144)

9.23.3.36 int e_po3030k_set_vsync (unsigned int start, unsigned int stop, unsigned int col)
Set the camera window VSYNC coordinate

**Warning:**
This is an internal function, use e_po3030k_ConfigCam

**Parameters:**
start The start row
stop The stop row
col The start/stop column

**Returns:**
Zero if OK, non-zero if an error occur

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

9.23.3.37 int e_po3030k_set_weight_win (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
Set the Weighting Window coordinate

**Parameters:**
x1 The X1 coordinate $\in \{211, x2\}$
x2 The X2 coordinate $\in \{x1 + 1, 423\}$
y1 The Y1 coordinate $\in \{160, y2\}$
y2 The Y2 coordinate $\in \{y1 + 1, 319\}$
9.23 camera/slow_3_timer/e_po3030k.h File Reference

Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p. 49

9.23.3.38  void e_po3030k_set_ww (unsigned char ww)
Set the Center weight (Back Light compensation) Control parameter

Parameters:
    ww  Center weight (4 lower bits only)

See also:
Datasheet p.44

9.23.3.39  int e_po3030k_set_wx (unsigned int start, unsigned int stop)
Set the camera window X coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
    start  The start column
    stop   The stop column

Returns:
Zero if OK, non-zero if an error occur

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

9.23.3.40  int e_po3030k_set_wy (unsigned int start, unsigned int stop)
Set the camera window Y coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
    start  The start row
    stop   The stop row
Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

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

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

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

Parameters:
start The beginning address of the write
stop The last write address

Returns:
The number of register written

See also:
e_po3030k_write_cam_registers (p. 187)

9.23.3.42 void e_po3030k_write_cam_registers (void)

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

See also:
e_po3030k_read_cam_registers (p. 177)

9.23.3.43 void e_po3030k_write_gamma_coef (void)

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

Warning:
This function need extra care from the user

See also:
e_po3030k_set_gamma_coef (p. 181)
Manage po3030k registers (two timers).

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

Defines

- `#define MCLK ((long) 14745600)
- `#define MCLK_P_NS 0.067816840278
- `#define ARRAY_ORIGINE_X 210
- `#define ARRAY_ORIGINE_Y 7
- `#define BASE_D1 0
- `#define BASE_D2 0
- `#define BASE_D3 0
- `#define BASE_D4 0
- `#define DEVICE_ID 0xDC
- `#define FRAME_WIDTH 0x0353
- `#define FRAME_HEIGHT 0x01e8
- `#define WINDOW_X1_BASE 9
- `#define WINDOW_Y1_BASE 13
- `#define WINDOW_X2_BASE 17
- `#define WINDOW_Y2_BASE 21
- `#define BIAS_BASE 25
- `#define COLGAIN_BASE 29
- `#define INTEGR_BASE 39
- `#define SPEED_ADDR (45 - BASE_D1)
- `#define MIRROR_BASE 47
- `#define SAMPLING_ADDR (51 - BASE_D2)
- `#define ADCOFF_BASE (53 - BASE_D2)
- `#define FLICKM_BASE (55 - BASE_D2)
- `#define FLICKP_BASE 59
- `#define COLOR_M_ADDR (67 - BASE_D3)
- `#define MODE_R5G6B5 0x08
- `#define MODE_YUV 0x02
- `#define MODE_GRAYSCALE 0x0c
- `#define SEPIA_BASE 69
- `#define LENS_C_BASE 73
- `#define EDGE_BASE 79
- `#define GAMMA_BASE 83
- `#define COLOR_COEF_BASE 107
- `#define CBCRGAIN_BASE 125
- `#define BRICTRY_BASE 129
- `#define SEPIATONE_BASE 133
- `#define VSYNCSTART_BASE (145 - BASE_D4)
- `#define VSYNCSTOP_BASE (149 - BASE_D4)
- `#define VSYNCCOL_BASE (153 - BASE_D4)
- `#define WW_BASE 157
- `#define AWVAETOL_BASE 159
```
• #define AESPEED_BASE 161
• #define EXPOSURE_BASE 163
• #define REFREPO_BASE 169
• #define MINMAXEXP_BASE 175
• #define MINMAXAWB_BASE 183
• #define WEIGHWIN_BASE 195
• #define AWBAEENABLE_BASE 211
• #define GAMMASELCOL_BASE 215
• #define NB_REGISTERS (sizeof(cam_reg)/(2*sizeof(cam_reg[0])))

Functions

• void e_po3030k_write_cam_registers (void)
• void e_po3030k_read_cam_registers (void)
• int e_po3030k_set_color_mode (int mode)
• int e_po3030k_set_sampling_mode (int mode)
• int e_po3030k_set_speed (int mode)
• int e_po3030k_set_wx (unsigned int start, unsigned int stop)
• int e_po3030k_set_wy (unsigned int start, unsigned int stop)
• int e_po3030k_set_vsync (unsigned int start, unsigned int stop, unsigned int col)
• int e_po3030k_set_register (unsigned char adr, unsigned char value)
• int e_po3030k_get_register (unsigned char adr, unsigned char *value)
• void e_po3030k_set_bias (unsigned char pixbias, unsigned char opbias)
• int e_po3030k_set_color_gain (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
• void e_po3030k_set_integr_time (unsigned long time)
• void e_po3030k_set_mirror (int vertical, int horizontal)
• void e_po3030k_set_adc_offset (unsigned char offset)
• void e_po3030k_set_sepia (int status)
• void e_po3030k_set_lens_gain (unsigned char red, unsigned char green, unsigned char blue)
• void e_po3030k_set_edge_prop (unsigned char gain, unsigned char tresh)
• void e_po3030k_set_gamma_coef (unsigned char array[12], char color)
• void e_po3030k_write_gamma_coef (void)
• int e_po3030k_sync_register_array (unsigned char start, unsigned char stop)
• void e_po3030k_SetColorMatrix (unsigned char array[3 * 3])
• void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)
• void e_po3030k_set_brigh_contr (unsigned char cg11c, unsigned char cg22c)
• void e_po3030k_set_sephia_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_awb_ae_speed (unsigned char b, unsigned char d)
• void e_po3030k_set_exposure (long t)
• void e_po3030k_set_ref_exposure (unsigned char exp)
• void e_po3030k_set_max_min_exp (unsigned int max, unsigned int min)
• void e_po3030k_set_max_min_awb (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratiom, unsigned char ratiob)
• int e_po3030k_set_weight_win (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
• void e_po3030k_set_awb_ae (int awb, int ae)
• void e_po3030k_set_flicker_mode (int manual)
• void e_po3030k_set_flicker_detection (int hz50, int hz60)
• static long po3030k_get_pixelclock (void)
• int e_po3030k_set_flicker_man_set (int hz50, int hz60, int fdm, int fk, int tol)
9.24 Variables

- static unsigned char cam_reg []

9.24.1 Detailed Description

Manage po3030k registers (two timers).

Author:
Philippe Retornaz
9.24.2 Define Documentation

9.24.2.1 #define ADCOFF_BASE ( 53 - BASE_D2 )

9.24.2.2 #define AESPEED_BASE 161

9.24.2.3 #define ARRAY_ORIGINE_X 210

9.24.2.4 #define ARRAY_ORIGINE_Y 7

9.24.2.5 #define AWBAENABLE_BASE 211

9.24.2.6 #define AWVAETOL_BASE 159

9.24.2.7 #define BASE_D1 0

9.24.2.8 #define BASE_D2 0

9.24.2.9 #define BASE_D3 0

9.24.2.10 #define BASE_D4 0

9.24.2.11 #define BIAS_BASE 25

9.24.2.12 #define BRICTR_BASE 129

9.24.2.13 #define CBCRGAIN_BASE 125

9.24.2.14 #define COLGAIN_BASE 29

9.24.2.15 #define COLOR_COEF_BASE 107

9.24.2.16 #define COLOR_M_ADDR (67 - BASE_D3)

9.24.2.17 #define DEVICE_ID 0xDC

9.24.2.18 #define EDGE_BASE 79

9.24.2.19 #define EXPOSURE_BASE 163

9.24.2.20 #define FLICKM_BASE ( 55 - BASE_D2 )

9.24.2.21 #define FLICKP_BASE 59

9.24.2.22 #define FRAME_HEIGTH 0x01e8

9.24.2.23 #define FRAME_WIDTH 0x0353

9.24.2.24 #define GAMMA_BASE 83

9.24.2.25 #define GAMMASELCOL_BASE 215

9.24.2.26 #define INTEGR_BASE 39

9.24.2.27 #define LENSG_BASE 73

9.24.2.28 #define MCLK ((long) 14745600)

9.24.2.29 #define MCLK_P_NS 0.067816840278

9.24.2.30 #define MINMAXAWB_BASE 183
Parameters:
   \textit{adr} The address
   \textit{value} The pointer to the value to write to

Returns:
   Zero if register found, non-zero if not found

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

See also:
   \texttt{e_po3030k_set_register} (p. 183)

\subsection{9.24.3.2 \texttt{void e_po3030k_read_cam_registers (void)}}
Read the camera register

See also:
   \texttt{e_po3030k_write_cam_registers} (p. 187)

\subsection{9.24.3.3 \texttt{void e_po3030k_set_adc_offset (unsigned char offset)}}
Set the Analog to Digital Converter offset

Parameters:
   \textit{offset} The offset

See also:
   Datasheet p.28

\subsection{9.24.3.4 \texttt{void e_po3030k_set_ae_speed (unsigned char \textit{b}, unsigned char \textit{d})}}
Set AE speed

Parameters:
   \textit{b} AE speed factor when exposure time is decreasing (4 lower bits only)
   \textit{d} AE speed factor when exposure time is increasing (4 lower bits only)

See also:
   Datasheet p.44
9.24.3.5  void e_po3030k_set_awb_ae (int awb, int ae)

Enable/Disable AWB and AE

Parameters:

- **awb**: 1 means AWB enabled, 0 means disabled
- **ae**: 1 means AE enabled, 0 means disabled

See also:

- Datasheet p. 50

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

Set AWB/AE tolerance margin

Parameters:

- **awbm**: AWV Margin (4 lower bits only)
- **aem**: AW Margin (4 lower bits only)

See also:

- Datasheet p.44

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

Set the Pixel and amplificator bias. Increasing the bias produces better image quality, but increases camera’s power consumption.

Parameters:

- **pixbias**: The pixel bias
- **opbias**: The Amplificator bias

See also:

- Datasheet p.22

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

Set the Brightness & Contrast

Parameters:

- **bright**: The Brightness (signed 1+7bits fixed point format)
- **contrast**: The Contrast

See also:

- Datasheet p. 40
9.24.3.9  void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)

Set The color gain (Cb/Cr)

Parameters:


See also:

  Datasheet p. 40

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

Set the gains of the camera

Parameters:

  global  The global gain ∈ {0, 79}
  red     The red pixel’s gain (fixed point [2:6] format)
  green1  The green pixel near read one gain ([2:6] format)
  green2  The green pixel near blue one gain ([2:6] format)
  blue    The blue pixel’s gain ([2:6] format)

Returns:

  Zero if OK, non-zero if an error occur

See also:

  Datasheet p.23-24

9.24.3.11 int e_po3030k_set_color_mode (int mode)

Set the camera color mode

Warning:

  This is an internal function, use e_po3030k_config_cam

Parameters:

  mode    The color mode

Returns:

  Zero if OK, non-zero if an error occur

See also:

  Datasheet p. 31, e_po3030k_write_cam_registers (p. 187) and e_po3030k_config_cam (p. 144)
9.24.3.12  void e_po3030k_set_edge_prop (unsigned char gain, unsigned char thresh)
Set Edge properties

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

  *thresh*  Edge Enhancement threshold

See also:
  
  Datasheet p.36

9.24.3.13  void e_po3030k_set_exposure (long t)
Set exposure time

Parameters:
  
  *t*  Exposure time, LSB is in 1/64 line time

Warning:
  
  Only writable if AE is disabled

See also:
  
  Datasheet p.45

9.24.3.14  void e_po3030k_set_flicker_detection (int hz50, int hz60)
Set the 50/60Hz flicker detection

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

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

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

See also:
  
  Datasheet p.29 e_po3030k_set_flicker_mode() (p. 134) e_po3030k_set_flicker_man_set() (p. 133)

9.24.3.15  int e_po3030k_set_flicker_man_set (int hz50, int hz60, int fdm, int fk, int tol)
Set the camera’s manual flicker’s detection setting

Parameters:
  
  *hz50*  The Hz for the 50Hz detection
hz60  The Hz for the 60Hz detection
fdm   Flicker duration mode
fk    Flicker count step
tol   Flicker tolerance

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

Returns:
Non-zero if an error occur, 0 if OK

See also:
Datasheet p.29-30  e_po3030k_set_flicker_detection() (p. 133)  e_po3030k_set_flicker_mode()
(p. 134)

9.24.3.16  void e_po3030k_set_flicker_mode (int manual)

Set flicker detection mode

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

See also:
Datasheet p.29  e_po3030k_set_flicker_detection() (p. 133)  e_po3030k_set_flicker_man_set()
(p. 133)

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

Set gamma coefficient

Warning:
This feature need extra care from the user

Parameters:
array  Gamma coefficient array
color  First two bytes : - 0b01 => Green
       • 0b00 => Red
       • else => Blue

See also:
Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef
9.24.3.18  void e_po3030k_set_integr_time (unsigned long time)

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

Parameters:

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

See also:

  Datasheet p.25

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

Set lens shading gain.

Parameters:

  *red*  Lens gain for red pixel ∈ \{0, 15\}
  
  *green*  Lens gain for green pixel ∈ \{0, 15\}
  
  *blue*  Lens gain for blue pixel ∈ \{0, 15\}

See also:

  Datasheet p.36

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

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

Parameters:

  *minb*  The minimum blue gain
  
  *maxb*  The maximum blue gain
  
  *minr*  The minimum red gain
  
  *maxr*  The maximum red gain
  
  *ratior*  The red gain ratio
  
  *ratiob*  The blue gain ratio

See also:

  Datasheet p. 47-48
9.24.3.21 void e_po3030k_set_max_min_exp (unsigned int max, unsigned int min)

Set the minimum and maximum exposure time in AE mode

Parameters:

min  The minimum exposure time
max  The maximum exposure time

See also:

Datasheet p.46-47

9.24.3.22 void e_po3030k_set_mirror (int vertical, int horizontal)

Enable/Disable horizontal or vertical mirror

Parameters:

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

See also:

Datasheet p.27

9.24.3.23 void e_po3030k_set_ref_exposure (unsigned char exp)

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

Parameters:

exp  The target exposure level

See also:

Datasheet p.45

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

Set the register adr to value value

Parameters:

adr  The address
value  The value

Returns:

Zero if register found, non-zero if not found

Warning:

This function is sub-optimal, if you use it heavy add an internal function to register.c
See also:
   e_po3030k_get_register (p. 176)

9.24.3.25  int e_po3030k_set_sampling_mode (int mode)

Set the camera sampling mode

Warning:
   This is an internal function, use e_po3030k_config_cam

Parameters:
   mode  The given sampling mode

Returns:
   Zero if OK, non-zero if an error occur

See also:
   Datasheet p. 28 and e_po3030k_config_cam (p. 144)

9.24.3.26  void e_po3030k_set_sepia (int status)

Enable/Disable Sepia color

Parameters:
   status  Set status to 1 to enable, 0 to disable

See also:
   Datasheet p.34 and 74

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

Set The color tone at sepia color condition

Parameters:
   cb  Cb tone
   cr  Cr tone

See also:
   e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74
9.24.3.28 int e_po3030k_set_speed (int mode)

Set the camera speed

Warning:
This is an internal function, use e_po3030k_config_cam

Parameters:
mode The given speed

Returns:
Zero if OK, non-zero if unknow mode

See also:
Datasheet p. 26 and e_po3030k_config_cam (p. 144)

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

Set the camera window VSYNC coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
start The start row
stop The stop row
col The start/stop column

Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p.42-43, e_po3030k_write_cam_registers (p. 187) and e_po3030k_config_cam (p. 144)

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

Set the Weighting Window coordinate

Parameters:
x1 The X1 coordinate ∈ \{211, x2\}
x2 The X2 coordinate ∈ \{x1 + 1, 423\}
y1 The Y1 coordinate ∈ \{160, y2\}
y2 The Y2 coordinate ∈ \{y1 + 1, 319\}
Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p. 49

9.24.3.31 void e_po3030k_set_ww (unsigned char ww)

Set the Center weight (Back Light compensation) Control parameter

Parameters:
ww Center weight ( 4 lower bits only )

See also:
Datasheet p.44

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

Set the camera window X coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
start The start column
stop The stop column

Returns:
Zero if OK, non-zero if an error occur

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

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

Set the camera window Y coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
start The start row
stop The stop row
Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

9.24.3.34 void e_po3030k_SetColorMatrix (unsigned char array[3 *3])
Set color correction coefficient

Parameters:

See also:
Datasheet p. 39

9.24.3.35 int e_po3030k_sync_register_array (unsigned char start, unsigned char stop)
Write every known register between address start and stop (inclusivly).

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

Parameters:
start The beginning address of the write
stop The last write address

Returns:
The number of register written

See also:
e_po3030k_write_cam_registers (p. 187)

9.24.3.36 void e_po3030k_write_cam_registers (void)
The Po3030k module keep in memory the state of each register the camera has. When you configure the camera, it only alter the internal register state, not the camera one. This function write the internal register state in the camera.

See also:
e_po3030k_read_cam_registers (p. 177)
9.24.3.37 void e_po3030k_write_gamma_coef (void)

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

Warning:

This function need extra care from the user

See also:

e_po3030k_set_gamma_coef (p. 181)

9.24.3.38 static long po3030k_get_pixelclock (void) [static]

9.24.4 Variable Documentation

9.24.4.1 unsigned char cam_reg[] [static]
Manage po3030k registers (three timers).
#include "./../I2C/e_I2C_protocol.h"
#include "e_po3030k.h"

Defines

- #define MCLK ((long) 14745600)
- #define MCLK_P_NS 0.067816840278
- #define ARRAY_ORIGINE_X 210
- #define ARRAY_ORIGINE_Y 7
- #define BASE_D1 0
- #define BASE_D2 0
- #define BASE_D3 0
- #define BASE_D4 0
- #define DEVICE_ID 0xDC
- #define FRAME_WIDTH 0x0353
- #define FRAME_HEIGHT 0x01e8
- #define WINDOW_X1_BASE 9
- #define WINDOW_Y1_BASE 13
- #define WINDOW_X2_BASE 17
- #define WINDOW_Y2_BASE 21
- #define BIAS_BASE 25
- #define COLGAIN_BASE 29
- #define INTEGR_BASE 39
- #define SPEED_ADDR (45 - BASE_D1)
- #define MIRROR_BASE 47
- #define SAMPLING_ADDR (51 - BASE_D2)
- #define ADCOFF_BASE (53 - BASE_D2)
- #define FLICKM_BASE (55 - BASE_D2)
- #define FLICKP_BASE 59
- #define COLOR_M_ADDR (67 - BASE_D3)
- #define MODE_R5G6B5 0x08
- #define MODE_YUV 0x02
- #define MODE_GRAYSCALE 0x0c
- #define SEPIA_BASE 69
- #define LENSG_BASE 73
- #define EDGE_BASE 79
- #define GAMMA_BASE 83
- #define COLOR_COEF_BASE 107
- #define CBCRGAIN_BASE 125
- #define BRICTR_BASE 129
- #define SEPIATONE_BASE 133
- #define VSYNCSTART_BASE (145 - BASE_D4)
- #define VSYNCTSTOP_BASE (149 - BASE_D4)
- #define VSYNCCOL_BASE (153 - BASE_D4)
- #define WW_BASE 157
- #define AWVAETOL_BASE 159
• #define AESPEED_BASE 161
• #define EXPOSURE_BASE 163
• #define REFREPO_BASE 169
• #define MINMAXEXP_BASE 175
• #define MINMAXAWB_BASE 183
• #define WEIGHWIN_BASE 195
• #define AWBAEENABLE_BASE 211
• #define GAMMASELCOL_BASE 215
• #define NB_REGISTERS (sizeof(cam_reg)/(2*sizeof(cam_reg[0])))

Functions

• void e_po3030k_write_cam_registers (void)
• void e_po3030k_read_cam_registers (void)
• int e_po3030k_set_color_mode (int mode)
• int e_po3030k_set_sampling_mode (int mode)
• int e_po3030k_set_speed (int mode)
• int e_po3030k_set_wx (unsigned int start, unsigned int stop)
• int e_po3030k_set_wy (unsigned int start, unsigned int stop)
• int e_po3030k_set_vsync (unsigned int start, unsigned int stop, unsigned int col)
• int e_po3030k_set_register (unsigned char adr, unsigned char value)
• int e_po3030k_get_register (unsigned char adr, unsigned char *value)
• void e_po3030k_set_bias (unsigned char pixbias, unsigned char opbias)
• int e_po3030k_set_color_gain (unsigned char global, unsigned char red, unsigned char green1, unsigned char green2, unsigned char blue)
• void e_po3030k_set_integr_time (unsigned long time)
• void e_po3030k_set_mirror (int vertical, int horizontal)
• void e_po3030k_set_adc_offset (unsigned char offset)
• void e_po3030k_set_sepia (int status)
• void e_po3030k_set_lens_gain (unsigned char red, unsigned char green, unsigned char blue)
• void e_po3030k_set_edge_prop (unsigned char gain, unsigned char tresh)
• void e_po3030k_set_gamma_coef (unsigned char array[12], char color)
• void e_po3030k_write_gamma_coef (void)
• int e_po3030k_sync_register_array (unsigned char start, unsigned char stop)
• void e_po3030k_setColorMatrix (unsigned char array[3 * 3])
• void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)
• void e_po3030k_set_brigh_contr (unsigned char brigh, unsigned char contr)
• 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_awb_ae_speed (unsigned char b, unsigned char d)
• void e_po3030k_set_exposure (long t)
• void e_po3030k_set_ref_exposure (unsigned char exp)
• void e_po3030k_set_max_min_exp (unsigned int max, unsigned int min)
• void e_po3030k_set_max_min_awb (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratiom, unsigned char ratiob)
• int e_po3030k_set_weight_win (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)
• void e_po3030k_set_awb_ae (int awb, int ae)
• void e_po3030k_set_flicker_mode (int manual)
• void e_po3030k_set_flicker_detection (int hz50, int hz60)
• static long po3030k_get_pixelclock (void)
• int e_po3030k_set_flicker_man_set (int hz50, int hz60, int fdm, int fk, int tol)
Variables

• static unsigned char cam_reg [ ]

9.25.1 Detailed Description

Manage po3030k registers (three timers).

Author:

Philippe Retornaz
9.25.2 Define Documentation

9.25.2.1 #define ADCOFF_BASE ( 53 - BASE_D2 )

9.25.2.2 #define AESPEED_BASE 161

9.25.2.3 #define ARRAY_ORIGINE_X 210

9.25.2.4 #define ARRAY_ORIGINE_Y 7

9.25.2.5 #define AWBAENABLE_BASE 211

9.25.2.6 #define AWVAETOL_BASE 159

9.25.2.7 #define BASE_D1 0

9.25.2.8 #define BASE_D2 0

9.25.2.9 #define BASE_D3 0

9.25.2.10 #define BASE_D4 0

9.25.2.11 #define BIAS_BASE 25

9.25.2.12 #define BRICTR_BASE 129

9.25.2.13 #define CBCRGAIN_BASE 125

9.25.2.14 #define COLGAIN_BASE 29

9.25.2.15 #define COLOR_COEF_BASE 107

9.25.2.16 #define COLOR_M_ADDR (67 - BASE_D3)

9.25.2.17 #define DEVICE_ID 0xDC

9.25.2.18 #define EDGE_BASE 79

9.25.2.19 #define EXPOSURE_BASE 163

9.25.2.20 #define FLICKM_BASE ( 55 - BASE_D2 )

9.25.2.21 #define FLICKP_BASE 59

9.25.2.22 #define FRAME_HEIGTH 0x01e8

9.25.2.23 #define FRAME_WIDTH 0x0353

9.25.2.24 #define GAMMA_BASE 83

9.25.2.25 #define GAMMASELCOL_BASE 215

9.25.2.26 #define INTEGR_BASE 39

9.25.2.27 #define LENSG_BASE 73

9.25.2.28 #define MCLK ((long) 14745600)

9.25.2.29 #define MCLK_P_NS 0.067816840278

9.25.2.30 #define MINMAXA WB_BASE 183

9.25.2.31 #define MINMAXEXP_BASE 175

9.25.2.32 #define MIRROR_BASE 47

9.25.2.33 #define MODE_GRAYSCALE 0x0c

9.25.2.34 #define MODE_R5G6B5 0x08

9.25.2.35 #define MODE_YUV 0x02

9.25.2.36 #define NB_REGISTERS (sizeof(cam_reg)/(2

9.25.2.37 #define REFREPO_BASE 169

9.25.2.38 #define SAMPLING_ADDR ( 51 - BASE_D2 )

9.25.2.39 #define SEPIA_BASE 69

9.25.2.40 #define SEPIATONE_BASE 133

9.25.2.41 #define SPEED_ADDR (45 - BASE_D1)

9.25.2.42 #define VSYNCCOL_BASE (153 - BASE_D4)

9.25.2.43 #define VSYNCSTART_BASE (145 - BASE_D4)

9.25.2.44 #define VSYNCSTOP_BASE (149 - BASE_D4)

9.25.2.45 #define WEIGHWIN_BASE 195

9.25.2.46 #define WINDOW_X1_BASE 9

9.25.2.47 #define WINDOW_X2_BASE 17

9.25.2.48 #define WINDOW_Y1_BASE 13

9.25.2.49 #define WINDOW_Y2_BASE 21

9.25.2.50 #define WW_BASE 157
Parameters:

 adr  The address
 value  The pointer to the value to write to

Returns:

 Zero if register found, non-zero if not found

Warning:

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

See also:

e_po3030k_set_register (p. 183)

9.25.3.2  void e_po3030k_read_cam_registers (void)

Read the camera register

See also:

e_po3030k_write_cam_registers (p. 187)

9.25.3.3  void e_po3030k_set_adc_offset (unsigned char offset)

Set the Analog to Digital Converter offset

Parameters:

 offset  The offset

See also:

Datasheet p.28

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

Set AE speed

Parameters:

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

See also:

Datasheet p.44
Enable/Disable AWB and AE

Parameters:

- `awb` 1 mean AWB enabled, 0 mean disabled
- `ae` 1 mean AE enabled, 0 mean disabled

See also:

- Datasheet p. 50

Set AWB/AE tolerance margin

Parameters:

- `awbm` AWV Margin (4 lower bits only)
- `aem` AW Margin (4 lower bits only)

See also:

- Datasheet p. 44

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

Parameters:

- `pixbias` The pixel bias
- `opbias` The Amplificator bias

See also:

- Datasheet p. 22

Set the Brightness & Contrast

Parameters:

- `bright` The Brightness (signed 1+7bits fixed point format)
- `contrast` The Contrast

See also:

- Datasheet p. 40
9.25.3.9  void e_po3030k_set_cb_cr_gain (unsigned char cg11c, unsigned char cg22c)

Set The color gain (Cb/Cr)

Parameters:

    cg11c  Cb gain (Sign[7] | Integer[6:5] | fractional[4:0])
    cg22c  Cr gain (Sign[7] | Integer[6:5] | fractional[4:0])

See also:

    Datasheet p. 40

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

Set the gains of the camera

Parameters:

    global  The global gain ∈ \{0, 79\}
    red     The red pixel’s gain (fixed point [2:6] format)
    green1  The green pixel near read one gain ([2:6] format)
    green2  The green pixel near blue one gain ([2:6] format)
    blue    The blue pixel’s gain ([2:6] format)

Returns:

    Zero if OK, non-zero if an error occur

See also:

    Datasheet p.23-24

9.25.3.11  int e_po3030k_set_color_mode (int mode)

Set the camera color mode

Warning:

    This is an internal function, use e_po3030k_config_cam

Parameters:

    mode  The color mode

Returns:

    Zero if OK, non-zero if an error occur

See also:

    Datasheet p. 31, e_po3030k_write_cam_registers (p. 187) and e_po3030k_config_cam (p. 144)
9.25.3.12  void e_po3030k_set_edge_prop (unsigned char gain, unsigned char tresh)

Set Edge properties

Parameters:

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

See also:

  Datasheet p.36

9.25.3.13  void e_po3030k_set_exposure (long t)

Set exposure time

Parameters:

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

Warning:

  Only writable if AE is disabled

See also:

  Datasheet p.45

9.25.3.14  void e_po3030k_set_flicker_detection (int hz50, int hz60)

Set the 50/60Hz flicker detection

Parameters:

  hz50  Non-zero mean 50Hz flicker detection enabled (default disabled)
  hz60  Non-zero mean 60Hz flicker detection enabled (default disabled)

Warning:

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

See also:

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

9.25.3.15  int e_po3030k_set_flicker_man_set (int hz50, int hz60, int fdm, int fk, int tol)

Set the camera’s manual flicker’s detection setting

Parameters:

  hz50  The Hz for the 50Hz detection
hz60  The Hz for the 60Hz detection
fdm   Flicker duration mode
fk    Flicker count step
tol   Flicker tolerance

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

Returns:
Non-zero if an error occur, 0 if OK

See also:
Datasheet p.29-30  e_po3030k_set_flicker_detection() (p. 133)  e_po3030k_set_flicker_mode()
(p. 134)

9.25.3.16 void e_po3030k_set_flicker_mode (int manual)
Set flicker detection mode

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

See also:
Datasheet p.29  e_po3030k_set_flicker_detection() (p. 133)  e_po3030k_set_flicker_man_set()
(p. 133)

9.25.3.17 void e_po3030k_set_gamma_coef (unsigned char array[12], char color)
Set gamma coefficient

Warning:
This feature need extra care from the user

Parameters:
array  Gamma coefficient array
color  First two bytes: - 0b01 => Green
       * 0b00 => Red
       * else => Blue

See also:
Datasheet p. 38, 50, 57-58 and e_po3030k_WriteGammaCoef
9.25.3.18 void e_po3030k_set_integr_time (unsigned long time)

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

Parameters:

\textit{time} The integration time ( fixed point [14:6] format )

See also:

Datasheet p.25

9.25.3.19 void e_po3030k_set_lens_gain (unsigned char red, unsigned char green, unsigned char blue)

Set lens shading gain

Parameters:

\textit{red} Lens gain for red pixel \in \{0, 15\}
\textit{green} Lens gain for green pixel \in \{0, 15\}
\textit{blue} Lens gain for blue pixel \in \{0, 15\}

See also:

Datasheet p.36

9.25.3.20 void e_po3030k_set_max_min_awb (unsigned char minb, unsigned char maxb, unsigned char minr, unsigned char maxr, unsigned char ratiorm, unsigned char ratiob)

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

Parameters:

\textit{minb} The minimum blue gain
\textit{maxb} The maximum blue gain
\textit{minr} The minimum red gain
\textit{maxr} The maximum red gain
\textit{ratiorm} The red gain ratio
\textit{ratiob} The blue gain ratio

See also:

Datasheet p. 47-48
9.25.3.21  void e_po3030k_set_max_min_exp (unsigned int max, unsigned int min)

Set the minimum and maximum exposure time in AE mode

Parameters:

  min  The minimum exposure time
  max  The maximum exposure time

See also:

  Datasheet p.46-47

9.25.3.22  void e_po3030k_set_mirror (int vertical, int horizontal)

Enable/Disable horizontal or vertical mirror

Parameters:

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

See also:

  Datasheet p.27

9.25.3.23  void e_po3030k_set_ref_exposure (unsigned char exp)

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

Parameters:

  exp  The target exposure level

See also:

  Datasheet p.45

9.25.3.24  int e_po3030k_set_register (unsigned char adr, unsigned char value)

Set the register \textit{adr} to value \textit{value}

Parameters:

  adr  The address
  value  The value

Returns:

  Zero if register found, non-zero if not found

Warning:

  This function is sub-optimal, if you use it heavily add an internal function to register.c
See also:

e_po3030k_get_register (p. 176)

9.25.3.25  int e_po3030k_set_sampling_mode (int mode)

Set the camera sampling mode

Warning:

This is an internal function, use e_po3030k_config_cam

Parameters:

mode  The given sampling mode

Returns:

Zero if OK, non-zero if an error occur

See also:

Datasheet p. 28 and e_po3030k_config_cam (p. 144)

9.25.3.26  void e_po3030k_set_sepia (int status)

Enable/Disable Sepia color

Parameters:

status  Set status to 1 to enable, 0 to disable

See also:

Datasheet p.34 and 74

9.25.3.27  void e_po3030k_set_sepia_tone (unsigned char cb, unsigned char cr)

Set The color tone at sepia color condition

Parameters:

cb  Cb tone

cr  Cr tone

See also:

e_po3030k_set_sepia (p. 184) and Datasheet p. 41 and 74
9.25.3.28 int e_po3030k_set_speed (int mode)

Set the camera speed

**Warning:**

This is an internal function, use e_po3030k_config_cam

**Parameters:**

*mode* The given speed

**Returns:**

Zero if OK, non-zero if unknown mode

**See also:**

Datasheet p. 26 and e_po3030k_config_cam (p. 144)

9.25.3.29 int e_po3030k_set_vsync (unsigned int start, unsigned int stop, unsigned int col)

Set the camera window VSYNC coordinate

**Warning:**

This is an internal function, use e_po3030k_ConfigCam

**Parameters:**

*start* The start row
*stop* The stop row
*col* The start/stop column

**Returns:**

Zero if OK, non-zero if an error occur

**See also:**

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

9.25.3.30 int e_po3030k_set_weight_win (unsigned int x1, unsigned int x2, unsigned int y1, unsigned int y2)

Set the Weighting Window coordinate

**Parameters:**

*x1* The X1 coordinate ∈ \{211, x2\}
*x2* The X2 coordinate ∈ \{x1 + 1, 423\}
*y1* The Y1 coordinate ∈ \{160, y2\}
*y2* The Y2 coordinate ∈ \{y1 + 1, 319\}
Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p. 49

9.25.3.31 void e_po3030k_set_ww (unsigned char ww)
Set the Center weight (Back Light compensation) Control parameter

Parameters:
ww Center weight ( 4 lower bits only )

See also:
Datasheet p.44

9.25.3.32 int e_po3030k_set_wx (unsigned int start, unsigned int stop)
Set the camera window X coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
start The start column
stop The stop column

Returns:
Zero if OK, non-zero if an error occur

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

9.25.3.33 int e_po3030k_set_wy (unsigned int start, unsigned int stop)
Set the camera window Y coordinate

Warning:
This is an internal function, use e_po3030k_ConfigCam

Parameters:
start The start row
stop The stop row
Returns:
Zero if OK, non-zero if an error occur

See also:
Datasheet p.20-21, e_po3030k_WriteCamRegisters, e_po3030k_SetWX and e_po3030k_ConfigCam

9.25.3.34 void e_po3030k_SetColorMatrix (unsigned char array[3 *3])
Set color correction coefficient

Parameters:

See also:
Datasheet p. 39

9.25.3.35 int e_po3030k_sync_register_array (unsigned char start, unsigned char stop)
Write every known register between address start and stop (inclusively).

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

Parameters:
start The beginning address of the write
stop The last write address

Returns:
The number of register written

See also:
e_po3030k_write_cam_registers (p. 187)

9.25.3.36 void e_po3030k_write_cam_registers (void)
The Po3030k module keep in memory the state of each register the camera has. When you configure the camera, it only alter the internal register state, not the camera one. This function write the internal register state in the camera.

See also:
e_po3030k_read_cam_registers (p. 177)
9.25.3.37 void e_po3030k_write_gamma_coef (void)

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

Warning:

This function need extra care from the user

See also:

   e_po3030k_set_gamma_coef (p. 181)

9.25.3.38 static long po3030k_get_pixelclock (void) [static]

9.25.4 Variable Documentation

9.25.4.1 unsigned char cam_reg[ ] [static]
9.26 camera/fast_2_timer/e_timers.c File Reference

Manage camera’s interrupts (two timers).

```c
#include "p30f6014A.h"
#include "../../../motor_led/e_epuck_ports.h"
#include "e_po3030k.h"
```

### Functions

- `void __attribute__ ((interrupt, auto_psv))
  The VSYNC interrupt. This interrupt is called every time the Vertical sync signal is asserted. This means that the picture is coming from the camera (we will have the first line soon).`

- `static void init_timer5 (void)`
- `static void init_timer4 (void)`
- `void e_po3030k_launch_capture (char *buf)`
- `int e_po3030k_apply_timer_config (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)`
- `int e_po3030k_is_img_ready (void)`

### Variables

- `char * po3030k_buffer`
- `int po3030k_img_ready`
- `static int blank_row_betw`
- `int po3030k_current_row`
- `int po3030k_row`
- `char po3030k_line_conf [330]`

#### 9.26.1 Detailed Description

Manage camera’s interrupts (two timers).

**Author:**

Philippe Retornaz

#### 9.26.2 Function Documentation

##### 9.26.2.1 void __attribute__ ((interrupt, auto_psv))

The VSYNC interrupt. This interrupt is called every time the Vertical sync signal is asserted. This means that the picture is coming from the camera (we will have the first line soon).

The Pixel Clock interrupt. This interrupt is called every time the Pixel clock signal is asserted. This means that the next byte is ready to be read.
9.26.2.2  int e_po3030k_apply_timer_config (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)

Modify the interrupt configuration

Warning:
This is an internal function, use e_po3030k_config_cam

Parameters:

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

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

See also:
e_po3030k_get_bytes_per_pixel (p. 145) and e_po3030k_config_cam (p. 144)

9.26.2.3  int e_po3030k_is_img_ready (void)

Check if the current capture is finished

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

See also:
e_po3030k_launch_capture (p. 193)

9.26.2.4  void e_po3030k_launch_capture (char * buf)

Launch a capture in the **buf** buffer

Parameters:

- **buf** The buffer to write to

See also:
e_po3030k_config_cam (p. 144) and e_po3030k_is_img_ready (p. 193)
9.26.2.5  static void init_timer4 (void)  [static]

9.26.2.6  static void init_timer5 (void)  [static]

9.26.3  Variable Documentation

9.26.3.1  char∗ _po3030k_buffer

The buffer to write to

9.26.3.2  int _po3030k_current_row

9.26.3.3  int _po3030k_img_ready

The flag to tell, the image is ready or not Zero mean capture is in progress, non-zero mean capture done.

See also:

e_po3030k_is_img_ready (p. 193)

9.26.3.4  char _po3030k_line_conf[330]

9.26.3.5  int _po3030k_row

9.26.3.6  int blank_row_betw  [static]
9.27 camera/slow_3_timer/e_timers.c File Reference

Manage camera’s interrupts (three timers).

```c
#include <p30f6014a.h>
#include "./../../motor_LED/e_epuck_ports.h"
#include "e_po3030k.h"
```

**Functions**

- `void __attribute__((interrupt, auto_psv)) The HSYNC interrupt. This interrupt is called each time the Horizontal sync signal is asserted This mean we begin the a line of the picture.

- `static void init_timer5 (void)`
- `static void init_timer4 (void)`
- `static void init_timer1 (void)`
- `void e_po3030k_launch_capture (char *buf)`
- `void e_po3030k_apply_timer_config (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)`
- `int e_po3030k_is_img_ready (void)`

**Variables**

- `static int max_row`
- `static int max_col`
- `static char * buffer`
- `static int img_ready`
- `static int current_row`
- `static int current_col`
- `static int buf_pos`
- `static int bytes_per_pixel`
- `static int bpp_current`
- `static int pixel_betw_pixel`
- `static int pbp_current`
- `static int blank_betw_lines`
- `static int bbl_current`

9.27.1 Detailed Description

Manage camera’s interrupts (three timers).

9.27.2 Function Documentation

9.27.2.1 `void __attribute__((interrupt, auto_psv))`

The HSYNC interrupt. This interrupt is called each time the Horizontal sync signal is asserted This mean we begin the a line of the picture.
The Pixel Clock interrupt. This interrupt is called every time the Pixel clock signal is asserted. This means that the next byte is ready to be read.

The VSYNC interrupt. This interrupt is called every time the Vertical sync signal is asserted. This means that the picture is coming from the camera (we will have the first line soon).

9.27.2.2  void e_po3030k_apply_timer_config (int pixel_row, int pixel_col, int bpp, int pbp, int bbl)

Modify the interrupt configuration

**Warning:**

This is an internal function, use `e_po3030k_config_cam`

**Parameters:**

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

See also:

- `e_po3030k_get_bytes_per_pixel` (p. 145) and `e_po3030k_config_cam` (p. 144)

9.27.2.3  int e_po3030k_is_img_ready (void)

Check if the current capture is finished

**Returns:**

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

See also:

- `e_po3030k_launch_capture` (p. 193)

9.27.2.4  void e_po3030k_launch_capture (char * buf)

Launch a capture in the `buf` buffer

**Parameters:**

- `buf`  The buffer to write to

See also:

- `e_po3030k_config_cam` (p. 144) and `e_po3030k_is_img_ready` (p. 193)
9.27.2.5 static void init_timer1 (void) [static]
9.27.2.6 static void init_timer4 (void) [static]
9.27.2.7 static void init_timer5 (void) [static]

9.27.3 Variable Documentation

9.27.3.1 int bbl_current [static]
The current row we ignore between each effective row

9.27.3.2 int blank_betw_lines [static]
The number of blank row between each effective row

9.27.3.3 int bpp_current [static]
The current byte we are inside a pixel

9.27.3.4 int buf_pos [static]
Counter which is incremented each time we acquire a byte

9.27.3.5 char* buffer [static]
The buffer to write to

9.27.3.6 int bytes_per_pixel [static]
Number of bytes per pixel

9.27.3.7 int current_col [static]
The current column we are

9.27.3.8 int current_row [static]
The current row we are

9.27.3.9 int img_ready [static]
The flag to tell, the image is ready or not Zero mean capture is in progress, non-zero mean capture done.

See also:
  e_po3030k_is_img_ready (p. 193)
9.27.3.10 int max_col [static]
The number of bytes per row we should take

See also:
   e_po3030k_get_bytes_per_pixel (p. 145), e_po3030k_apply_timer_config (p. 193)

9.27.3.11 int max_row [static]
The number of row we should take

See also:
   e_po3030k_aApply_timer_config

9.27.3.12 int pixel_current [static]
The current pixel we are between two effective pixel

9.27.3.13 int pixel_betw_pixel [static]
Pixel to "jump" between each effective pixel
9.28  codec/e_common.inc File Reference
9.29 codec/e_sound.c File Reference

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

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

Functions

- void e_init_sound (void)
  
  Initialize all you need to play sound on speaker.

- void e_play_sound (int sound_nbr, int sound_length)
  
  Play a sound.

- void e_close_sound (void)
  
  Disable the sound After that you can’t play sound anymore, if you want to, you have to call e_init_sound.

9.29.1 Detailed Description

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

Author:

  Code: Michael Bonani
  Doc: Jonathan Besuchet

9.29.2 Function Documentation

9.29.2.1 void e_close_sound (void)

Disable the sound After that you can’t play sound anymore, if you want to, you have to call e_init_sound.

9.29.2.2 void e_init_sound (void)

Initialize all you need to play sound on speaker.

Warning:

  You must to call this function before playing a sound (call it only one time).

9.29.2.3 void e_play_sound (int sound_nbr, int sound_length)

Play a sound.
Parameters:

- `sound_nbr` the beginning of the sound
- `sound_length` the length of the sound

See also:

- `Sound` (p. 21)
9.30 codec/e_sound.h File Reference

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

Functions

• void e_init_sound (void)
  
  Initialize all you need to play sound on speaker.

• void e_play_sound (int sound_offset, int sound_length)

  Play a sound.

• void e_close_sound (void)

  Disable the sound After that you can’t play sound anymore, if you want to, you have to call e_init_sound.

• void e_init_dci_master (void)
• void e_init_codec_slave (void)
• void e_sub_dci_kickoff (int, int)

9.30.1 Detailed Description

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

Author:

  Code: Michael Bonani
  Doc: Jonathan Besuchet

9.30.2 Function Documentation

9.30.2.1 void e_close_sound (void)

Disable the sound After that you can’t play sound anymore, if you want to, you have to call e_init_sound.

9.30.2.2 void e_init_codec_slave (void)

9.30.2.3 void e_init_dci_master (void)

9.30.2.4 void e_init_sound (void)

Initialize all you need to play sound on speaker.

Warning:

  You must to call this function before playing a sound (call it only one time).
9.30.2.5  void e_play_sound (int sound_nbr, int sound_length)

Play a sound.

Parameters:

- `sound_nbr` the beginning of the sound
- `sound_length` the length of the sound

See also:

Sound (p. 21)

9.30.2.6  void e_sub_dci_kickoff (int, int)
#include "e_devantech.h"
#include "/../I2C/e_I2C_master_module.h"
#include "/../I2C/e_I2C_protocol.h"

Functions

- void e_init_devantech (void)
- void e_disable_devantech (void)
- unsigned int e_get_dist_devantech (char device_add)
- unsigned int e_get_delay_devantech (char device_add)
- char e_get_sr_devantech (char device_add)
- unsigned int e_get_light_devantech (char device_add)
- void e_set_gain_devantech (char device_add, char gain)
- void e_set_range_devantech (char device_add, char range)
- void e_i2cd_write (char device_add, char reg, char value)
- char e_i2cd_readb (char device_add, char reg)

9.31.1 Function Documentation

9.31.1.1 void e_disable_devantech (void)
9.31.1.2 unsigned int e_get_delay_devantech (char device_add)
9.31.1.3 unsigned int e_get_dist_devantech (char device_add)
9.31.1.4 unsigned int e_get_light_devantech (char device_add)
9.31.1.5 char e_get_sr_devantech (char device_add)
9.31.1.6 char e_i2cd_readb (char device_add, char reg)
9.31.1.7 void e_i2cd_write (char device_add, char reg, char value)
9.31.1.8 void e_init_devantech (void)
9.31.1.9 void e_set_gain_devantech (char device_add, char gain)
9.31.1.10 void e_set_range_devantech (char device_add, char range)
9.32 contrib/LIS_sensors_turret/e_devantech.h File Reference

Devantech sensor of e-puck.
#include "../../I2C/e_I2C_master_module.h"
#include "../../motor_led/e_epuck_ports.h"

Functions

• void e_init_devantech (void)

• void e_disable_devantech (void)

• unsigned int e_get_dist_devantech (char device_add)

• unsigned int e_get_delay_devantech (char device_add)

• char e_get_sr_devantech (char device_add)

• unsigned int e_get_light_devantech (char device_add)

• void e_set_gain_devantech (char device_add, char gain)

• void e_set_range_devantech (char device_add, char range)

• void e_i2cd_write (char device_add, char reg, char value)

• char e_i2cd_readb (char device_add, char reg)

• unsigned int e_i2cd_readw (char device_add, char reg)

9.32.1 Detailed Description

Devantech sensor of e-puck.

Author:
Jonathan Besuchet
### Function Documentation

#### 9.32.2.1 void e_disable_devantech (void)

#### 9.32.2.2 unsigned int e_get_delay_devantech (char device_add)

#### 9.32.2.3 unsigned int e_get_dist_devantech (char device_add)

#### 9.32.2.4 unsigned int e_get_light_devantech (char device_add)

#### 9.32.2.5 char e_get_sr_devantech (char device_add)

#### 9.32.2.6 char e_i2cd_readb (char device_add, char reg)

#### 9.32.2.7 unsigned int e_i2cd_readw (char device_add, char reg)

#### 9.32.2.8 void e_i2cd_write (char device_add, char reg, char value)

#### 9.32.2.9 void e_init_devantech (void)

#### 9.32.2.10 void e_set_gain_devantech (char device_add, char gain)

#### 9.32.2.11 void e_set_range_devantech (char device_add, char range)
# contrib/LIS_sensors_turret/e_sensext.c File Reference

```c
#include "e_sensext.h"
#include "e_sharp.h"
#include <a_d/advance_ad_scan/e_acc.h>
#include <motor_led/advance_one_timer/e_agenda.h>
#include <I2C/e_I2C_master_module.h>
#include <I2C/e_I2C_protocol.h>
```

## Functions

- `void e_stop_sensext_wait (void)`
- `void e_start_sensext_wait (void)`
- `int e_get_sensext_wait (void)`
- `void e_init_sensext (void)`
- `int e_sensext_process (int *sensext_param, unsigned int *sensext_value)`

## Variables

- `static int sensext_wait = 0`

### 9.33.1 Function Documentation

#### 9.33.1.1 int e_get_sensext_wait (void)

#### 9.33.1.2 void e_init_sensext (void)

#### 9.33.1.3 int e_sensext_process (int *sensext_param, unsigned int *sensext_value)

!!WALTER!!!!!

!!WALTER!!!!!

#### 9.33.1.4 void e_start_sensext_wait (void)

#### 9.33.1.5 void e_stop_sensext_wait (void)

### 9.33.2 Variable Documentation

#### 9.33.2.1 int sensext_wait = 0 [static]
9.34 contrib/LIS_sensors_turret/e_sensext.h File Reference

Defines

- `#define I2C_ADDR_SENSEXT 0b10100010`
- `#define I2C_ADDR_SRF08 0xE0`
- `#define I2C_ADDR_SRF10 0xE0`
- `#define I2C_ADDR_CMPS03 0xC0`
- `#define I2C_ADDR_SRF235 0xE0`

Functions

- `void e_init_sensext (void)`
- `int e_sensext_process (int *sensext_param, unsigned int *sensext_value)`
- `void e_stop_sensext_wait (void)`
- `void e_start_sensext_wait (void)`
- `int e_get_sensext_wait (void)`

9.34.1 Define Documentation

9.34.1.1 `#define I2C_ADDR_CMPS03 0xC0`
9.34.1.2 `#define I2C_ADDR_SENSEXT 0b10100010`
9.34.1.3 `#define I2C_ADDR_SRF08 0xE0`
9.34.1.4 `#define I2C_ADDR_SRF10 0xE0`
9.34.1.5 `#define I2C_ADDR_SRF235 0xE0`

9.34.2 Function Documentation

9.34.2.1 `int e_get_sensext_wait (void)`
9.34.2.2 `void e_init_sensext (void)`
9.34.2.3 `int e_sensext_process (int *sensext_param, unsigned int *sensext_value)`

!!WALTER!!!!!
!!WALTER!!!!!

9.34.2.4 `void e_start_sensext_wait (void)`
9.34.2.5 `void e_stop_sensext_wait (void)`
#include "a_d/e_ad_conv.h"
#include "motor_led/e_epuck_ports.h"
#include "e_sharp.h"
#include <a_d/advance_ad_scan/e_acc.h>

## Functions

- void `e_init_sharp` (void)
- int `e_get_dist_sharp` ()
- void `e_set_sharp_led` (unsigned int `sharp_led_number`, unsigned int `value`)
- void `e_sharp_led_clear` (void)
- void `e_sharp_on` (void)
- void `e_sharp_off` (void)

### 9.35.1 Function Documentation

#### 9.35.1.1 int e_get_dist_sharp (void)

#### 9.35.1.2 void e_init_sharp (void)

#### 9.35.1.3 void e_set_sharp_led (unsigned int `sharp_led_number`, unsigned int `value`)

#### 9.35.1.4 void e_sharp_led_clear (void)

#### 9.35.1.5 void e_sharp_off (void)

#### 9.35.1.6 void e_sharp_on (void)
9.36 contrib/LIS_sensors_turret/e_sharp.h File Reference

Defines

- #define SHARP 5
- #define SHARP_LED1 LATG6
- #define SHARP_LED2 LATG7
- #define SHARP_LED3 LATG8
- #define SHARP_LED4 LATG9
- #define SHARP_LED5 LATB6
- #define SHARP_VIN LATB7
- #define SHARP_LED1_DIR TRISG6
- #define SHARP_LED2_DIR TRISG7
- #define SHARP_LED3_DIR TRISG8
- #define SHARP_LED4_DIR TRISG9
- #define SHARP_LED5_DIR TRISB6
- #define SHARP_VIN_DIR TRISB7

Functions

- void e_init_sharp (void)
- int e_get_dist_sharp (void)
- void e_set_sharp_led (unsigned int sharp_led_number, unsigned int value)
- void e_sharp_led_clear (void)
- void e_sharp_on (void)
- void e_sharp_off (void)
9.36.1 Define Documentation

9.36.1.1 #define SHARP 5

9.36.1.2 #define SHARP_LED1 _LATG6

9.36.1.3 #define SHARP_LED1_DIR _TRISG6

9.36.1.4 #define SHARP_LED2 _LATG7

9.36.1.5 #define SHARP_LED2_DIR _TRISG7

9.36.1.6 #define SHARP_LED3 _LATG8

9.36.1.7 #define SHARP_LED3_DIR _TRISG8

9.36.1.8 #define SHARP_LED4 _LATG9

9.36.1.9 #define SHARP_LED4_DIR _TRISG9

9.36.1.10 #define SHARP_LED5 _LATB6

9.36.1.11 #define SHARP_LED5_DIR _TRISB6

9.36.1.12 #define SHARP_VIN _LATB7

9.36.1.13 #define SHARP_VIN_DIR _TRISB7

9.36.2 Function Documentation

9.36.2.1 int e_get_dist_sharp (void)

9.36.2.2 void e_init_sharp (void)

9.36.2.3 void e_set_sharp_led (unsigned int sharp_led_number, unsigned int value)

9.36.2.4 void e_sharp_led_clear (void)

9.36.2.5 void e_sharp_off (void)

9.36.2.6 void e_sharp_on (void)


```c
#include "ComModule.h"
#include <stdlib.h>
#include "./motor_led/e_epuck_ports.h"
#include "./I2C/e_I2C_protocol.h"
#include "./I2C/e_I2C_master_module.h"

Defines

- #define COM_MODULE_I2C_ADDR (0x3F<<1)
- #define BUFFER_DATA_LENGTH (COM_MODULE_MAXSIZE + 14)
- #define STATUS_REG_ADDR 0x00
- #define CONFIG_REG_ADDR 0x01
- #define SEND_REG_ADDR 0x02
- #define SOFTATT_REG_ADDR 0x03
- #define OWNGROUP_REG_ADDR 0x04
- #define OWNADDRL_REG_ADDR 0x05
- #define OWNADDRH_REG_ADDR 0x06
- #define SEND_BUFFER_START 0x07
- #define SEND_BUFFER_END (SEND_BUFFER_START + BUFFER_DATA_LENGTH)
- #define REC_BUFFER_START (SEND_BUFFER_END + 1)
- #define REC_BUFFER_END (REC_BUFFER_START + BUFFER_DATA_LENGTH)
- #define AM_MSGTYPE 0x0A
- #define AM_MSGTYPE_IN_PACKET_OFFSET 0x08
- #define ADDRLDATA_IN_PACKET_OFFSET 0x06
- #define ADDRHDATA_IN_PACKET_OFFSET 0x07
- #define TYPEDATA_IN_PACKET_OFFSET 0x08
- #define GROUPDATA_IN_PACKET_OFFSET 0x09
- #define FIRSTDATA_IN_PACKET_OFFSET 0x0A
- #define PACKET_READY_FLAG 0x01
- #define TX_IDLE_FLAG 0x02
- #define PACKET_LOST_FLAG 0x04
- #define TX_SEND_ERROR 0x08
- #define HARDWAREATT_SET_FLAG 0x01
- #define RADIO_ENABLED_FLAG 0x80
- #define REQUEST_TO_SEND_FLAG 0x01

Functions

- unsigned char ReadRegister (unsigned char registeraddr)
- void WriteRegister (unsigned char registeraddr, unsigned char value)
- void InitComModule (unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue)
- int IsModulePlugged ()
- void SetRadioEnabledState (unsigned char mode)
- void SetHardwareAttenuator (unsigned char attenuatormode)
- void SetSoftwareAttenuator (unsigned char attenuatorvalue)

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
• void **SetOwnGroup** (unsigned char owngroup)

• void **SetOwnAddress** (unsigned int ownaddress)

• unsigned char **GetHardwareAttenuator** ()

• unsigned char **GetRadioEnabledState** ()

• unsigned char **GetSoftwareAttenuator** ()

• unsigned char **GetOwnGroup** ()

• unsigned int **GetOwnAddress** ()

• unsigned char **GetStatus** ()

• int **SendPacket** (unsigned char destinationgroup, unsigned int destinationaddress, unsigned char *packet, int packetsize)

• int **IsPacketReady** (unsigned char *packet, int *packetsize)
9.37 Define Documentation

9.37.1 Define Documentation

9.37.1.1 #define ADDRHDATA_IN_PACKET_OFFSET 0x07

9.37.1.2 #define ADDRLDATA_IN_PACKET_OFFSET 0x06

9.37.1.3 #define AM_MSGTYPE 0x0A

9.37.1.4 #define AM_MSGTYPE_IN_PACKET_OFFSET 0x08

9.37.1.5 #define BUFFER_DATA_LENGTH (COM_MODULE_MAXSIZE + 14)

9.37.1.6 #define COM_MODULE_I2C_ADDR (0x3F << 1)

9.37.1.7 #define CONFIG_REG_ADDR 0x01

9.37.1.8 #define FIRSTDATA_IN_PACKET_OFFSET 0x0A

9.37.1.9 #define GROUPDATA_IN_PACKET_OFFSET 0x09

9.37.1.10 #define HARDWAREATT_SET_FLAG 0x01

9.37.1.11 #define OWNADDRH_REG_ADDR 0x06

9.37.1.12 #define OWNADDRL_REG_ADDR 0x05

9.37.1.13 #define OWNGROUP_REG_ADDR 0x04

9.37.1.14 #define PACKET_LOST_FLAG 0x04

9.37.1.15 #define PACKET_READY_FLAG 0x01

9.37.1.16 #define RADIO_ENABLED_FLAG 0x80

9.37.1.17 #define REC_BUFFER_END (REC_BUFFER_START + BUFFER_DATA_LENGTH)

9.37.1.18 #define REC_BUFFER_START (SEND_BUFFER_END + 1)

9.37.1.19 #define REQUEST_TO_SEND_FLAG 0x01

9.37.1.20 #define SEND_BUFFER_END (SEND_BUFFER_START + BUFFER_DATA_LENGTH)

9.37.1.21 #define SEND_BUFFER_START 0x07

9.37.1.22 #define SEND_REG_ADDR 0x02

9.37.1.23 #define SOFTATT_REG_ADDR 0x03

9.37.1.24 #define STATUS_REG_ADDR 0x00

9.37.1.25 #define TX_IDLE_FLAG 0x02

9.37.1.26 #define TX_SEND_ERROR 0x08

9.37.1.27 #define TYPEDATA_IN_PACKET_OFFSET 0x08

9.37.2 Function Documentation

9.37.2.1 unsigned char GetHardwareAttenuator ()

9.37.2.2 unsigned int GetOwnAddress ()

9.37.2.3 unsigned char GetOwnGroup ()

9.37.2.4 unsigned char GetRadioEnabledState ()

9.37.2.5 unsigned char GetSoftwareAttenuator ()

9.37.2.6 unsigned char GetStatus ()

9.37.2.7 void InitComModule (unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue)

9.37.2.8 int IsModulePlugged ()

9.37.2.9 int IsPacketReady (unsigned char∗ packet, int∗ packetsize)

9.37.2.10 unsigned char ReadRegister (unsigned char registeraddr)

9.37.2.11 int SendPacket (unsigned char destinationgroup, unsigned int destinationaddress, unsigned char∗ packet, int packetsize)

9.37.2.12 void SetHardwareAttenuator (unsigned char attenuatormode)

9.37.2.13 void SetOwnAddress (unsigned int ownaddress)

9.37.2.14 void SetOwnGroup (unsigned char owngroup)

9.37.2.15 void SetRadioEnabledState (unsigned char mode)

9.37.2.16 void SetSoftwareAttenuator (unsigned char attenuatorvalue)

9.37.2.17 void WriteRegister (unsigned char registeraddr, unsigned char value)
9.38 contrib/SWIS_com_module/ComModule.h File Reference

Radio communication.
#include "p30f6014A.h"

Defines

- #define COM_MODULE_HW_ATTENUATOR_25DB 1
- #define COM_MODULE_HW_ATTENUATOR_0DB 0
- #define COM_MODULE_DEFAULT_GROUP 0x7d
- #define COM_MODULE_MAXSIZE 108

Functions

- void InitComModule (unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue)
- int IsModulePlugged ()
- void SetRadioEnabledState (unsigned char mode)
- void SetHardwareAttenuator (unsigned char AttenuatorMode)
- void SetSoftwareAttenuator (unsigned char AttenuatorValue)
- void SetOwnGroup (unsigned char GroupID)
- void SetOwnAddress (unsigned int ownaddress)
- unsigned char GetRadioEnabledState ()
- unsigned char GetHardwareAttenuator ()
- unsigned char GetSoftwareAttenuator ()
- unsigned char GetOwnGroup ()
- unsigned char GetStatus ()
- int SendPacket (unsigned char destinationgroup, unsigned int destinationaddress, unsigned char ∗packet, int packetsize)
- int IsPacketReady (unsigned char ∗packet, int ∗packetSize)

9.38.1 Detailed Description

Radio communication.

Author:
Jonathan Besuchet
9.38.2 Define Documentation

9.38.2.1 #define COM_MODULE_DEFAULT_GROUP 0x7d

9.38.2.2 #define COM_MODULE_HW_ATTENUATOR_0DB 0

9.38.2.3 #define COM_MODULE_HW_ATTENUATOR_25DB 1

9.38.2.4 #define COM_MODULE_MAXSIZE 108

9.38.3 Function Documentation

9.38.3.1 unsigned char GetHardwareAttenuator ()

9.38.3.2 unsigned char GetOwnGroup ()

9.38.3.3 unsigned char GetRadioEnabledState ()

9.38.3.4 unsigned char GetSoftwareAttenuator ()

9.38.3.5 unsigned char GetStatus ()

9.38.3.6 void InitComModule (unsigned char owngroup, unsigned int ownaddress, unsigned char hardwareattenuatormode, unsigned char softwareattenuatorvalue)

9.38.3.7 int IsModulePlugged ()

9.38.3.8 int IsPacketReady (unsigned char * packet, int * packetSize)

9.38.3.9 int SendPacket (unsigned char destinationgroup, unsigned int destinationaddress, unsigned char * packet, int packetsize)

9.38.3.10 void SetHardwareAttenuator (unsigned char AttenuatorMode)

9.38.3.11 void SetOwnAddress (unsigned int ownaddress)

9.38.3.12 void SetOwnGroup (unsigned char GroupID)

9.38.3.13 void SetRadioEnabledState (unsigned char mode)

9.38.3.14 void SetSoftwareAttenuator (unsigned char AttenuatorValue)
9.39  fft/e_fft.c File Reference

Package to manage the FFT.
#include <dsp.h>
#include "e_fft.h"

Functions
- const fractcomplex twiddleFactors[FFT_BLOCK_LENGTH/2] __attribute__ ((space(auto_psv), aligned(FFT_BLOCK_LENGTH *2)))
- void e_doFFT_asm (fractcomplex *sigCmpx)
  Execute the FFT with dsPic special instructions.

9.39.1  Detailed Description

Package to manage the FFT.

Author:
Code & doc: Jonathan Besuchet

9.39.2  Function Documentation

9.39.2.1  const fractcomplex twiddleFactors[FFT_BLOCK_LENGTH/2] __attribute__ ((space(auto_psv), aligned(FFT_BLOCK_LENGTH *2)))

9.39.2.2  void e_doFFT_asm (fractcomplex * sigCmpx)

Execute the FFT with dsPic special instructions.

Parameters:
- sigCmpx  The pointer of the begining of the array on which you want to perform the FFT.
9.40 fft/e_fft.h File Reference

Package to manage the FFT.

```c
#include <dsp.h>
```

Defines

- `#define FFT_BLOCK_LENGTH 256`
- `#define LOG2_BLOCK_LENGTH 8`

Functions

- `void e_doFFT_asm (fractcomplex *sigCmpx)`
  
  Execute the FFT with dsPic special instructions.

9.40.1 Detailed Description

Package to manage the FFT.

In this package the FFT (fast fourier transform) is done with the special dsPic 30fxxxx instructions.

Before calling the `e_doFFT_asm()` (p. 214) function, you must to fill the sigCmpx array with your values.

This array is stocked in the Y memory of the dsPic.

Author:

Code & doc: Jonathan Besuchet

9.40.2 Define Documentation

9.40.2.1 `#define FFT_BLOCK_LENGTH 256`

9.40.2.2 `#define LOG2_BLOCK_LENGTH 8`

9.40.3 Function Documentation

9.40.3.1 `void e_doFFT_asm (fractcomplex * sigCmpx)`

Execute the FFT with dsPic special instructions.

Parameters:

`sigCmpx` The pointer of the begining of the array on which you want to perform the FFT.
9.41 fft/e_fft_utilities.h File Reference

Some fft features.

Functions

- void e_fast_copy (int *in_array, int *out_array, int size)
  
  Copy an array to an other array, using REPEAT instruction.

- void e_subtract_mean (int *array, int size, int log2size)
  
  Subtract the mean from the samples of the array (-> produce zero-mean samples).

9.41.1 Detailed Description

Some fft features.

Here you have two functions that are really useful to work with the FFT

Author:

Code & doc: Jonathan Besuchet

9.41.2 Function Documentation

9.41.2.1 void e_fast_copy (int *in_array, int *out_array, int size)

Copy an array to an other array, using REPEAT instruction.

Parameters:

- in_array The array from which you want to copy
- out_array The destination array
- size The number of element you want to copy

9.41.2.2 void e_subtract_mean (int *array, int size, int log2size)

Subtract the mean from the samples of the array (-> produce zero-mean samples).

Parameters:

- array The array that you want to produce zero-mean samples
- size The size of the array
- log2size The log in base 2 of the size

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
Allocate memory and initialize the sigCmpx array.

```c
#include <dsp.h>
#include "e_fft.h"
```

### Functions

- `fractcomplex sigCmpx[FFT_BLOCK_LENGTH] __attribute__((section(".ydata, data, ymemory"), aligned(FFT_BLOCK_LENGTH * 2 * 2)))`

### 9.42.1 Detailed Description

Allocate memory and initialize the sigCmpx array.

sigCmpx is the main array in which the FFT will be done. So you must fill this array with your values to perform the FFT.

This array has the following size: 64, 128, 256, 512 depending on the choice you have made in fft.h

**Author:**

Code & doc: Jonathan Besuchet

### 9.42.2 Function Documentation

#### 9.42.2.1 `fractcomplex sigCmpx [FFT_BLOCK_LENGTH] __attribute__((section(".ydata, data, ymemory"), aligned(FFT_BLOCK_LENGTH * 2 * 2)))`
9.43 fft/e_twiddle_factors.c File Reference

The FFT factor from Microchip.
#include <dsp.h>
#include "e_fft.h"

Functions

• const fractcomplex twiddleFactors[] __attribute__((space(auto_psv), aligned(FFT_BLOCK_LENGTH * 2)))

9.43.1 Detailed Description

The FFT factor from Microchip.

Author:
Jonathan Besuchet

9.43.2 Function Documentation

9.43.2.1 const fractcomplex twiddleFactors[] __attribute__((space(auto_psv), aligned(FFT_BLOCK_LENGTH + 2)))
Manage I2C basics.
#include "e_I2C_master_module.h"

Functions

- **void idle_i2c (void)**
  Wait until I2C Bus is Inactive.

- **char e_i2c_init (void)**
  Initialize the microcontroller for I2C uses.

- **char e_i2c_reset (void)**
  Reset the microcontroller for I2C uses.

- **char e_i2c_enable (void)**
  Enable special I2C interrupt.

- **char e_i2c_disable (void)**
  Disable special I2C interrupt.

- **char e_i2c_start (void)**
  Make the start bit.

- **char e_i2c_restart (void)**
  Make the restart bit.

- **char e_i2c_stop (void)**
  Make the stop bit.

- **char e_i2c_ack (void)**
  Make the acknowledgement bit.

- **char e_i2c_nack (void)**
  Make the non-acknowledgement bit.

- **char e_i2c_read (char *buf)**
  Read the I2C input register.

- **char e_i2c_write (char byte)**
  Write on the I2C output register.

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

Variables

- **char e_i2c_mode**
- **int e_interrupts [3]**
9.44.1 Detailed Description

Manage I2C basics.

This module manages the I2C basics functions (low level I2C functions).

They are made to perform the basics tasks like:

- initializing the I2C on the microcontroller
- sending the Start bit (e_i2c_start)
- sending the Restart bit (e_i2c_restart)
- sending the Stop bit (e_i2c_stop)
- sending the acknowledgement bit (e_i2c_ack)
- writing or receiving a byte (e_i2c_write, e_i2c_read)
- ...

Author:

Code: Davis Daidie
Doc: Jonathan Besuchet

9.44.2 Function Documentation

9.44.2.1 void __attribute__ ((__interrupt__, auto_psv))

9.44.2.2 char e_i2c_ack (void)

Make the acknowledgement bit.

Returns:

1 to confirm the operation and 0 for an error

9.44.2.3 char e_i2c_disable (void)

Disable special I2C interrupt.

Returns:

1 to confirm the operation and 0 for an error

9.44.2.4 char e_i2c_enable (void)

Enable special I2C interrupt.

Returns:

1 to confirm the operation and 0 for an error
9.44.2.5 char e_i2c_init (void)

Initialize the microcontroller for I2C uses.

Returns:
1 to confirm the operation and 0 for an error

9.44.2.6 char e_i2c_nack (void)

Make the non-acknowledgement bit.

Returns:
1 to confirm the operation and 0 for an error

9.44.2.7 char e_i2c_read (char *buf)

Read the I2C input register.

Parameters:
buf A pointer to store the datas received

Returns:
1 to confirm the operation and 0 for an error

9.44.2.8 char e_i2c_reset (void)

Reset the microcontroller for I2C uses.

Returns:
1 to confirm the operation and 0 for an error

9.44.2.9 char e_i2c_restart (void)

Make the restart bit.

Returns:
1 to confirm the operation and 0 for an error

9.44.2.10 char e_i2c_start (void)

Make the start bit.

Returns:
1 to confirm the operation and 0 for an error
9.44.2.11 char e_i2c_stop (void)

Make the stop bit.

Returns:
   1 to confirm the operation and 0 for an error

9.44.2.12 char e_i2c_write (char byte)

Write on the I2C output register.

Parameters:
   byte  What you want to send on I2C

Returns:
   1 to confirm the operation and 0 for an error

9.44.2.13 void idle_i2c (void)

Wait until I2C Bus is Inactive.

9.44.3 Variable Documentation

9.44.3.1 char e_i2c_mode

9.44.3.2 int e_interrupts[3]
Manage I2C basics.

```c
#include "p30f6014A.h"
```

### Defines

- `#define START 1`
- `#define WRITE 2`
- `#define ACKNOWLEDGE 3`
- `#define READ 4`
- `#define STOP 5`
- `#define RESTART 6`
- `#define ERROR 10`
- `#define OPERATION_OK 0`

### Functions

- `char e_i2c_init (void)`
  *Initialize the microcontroller for I2C uses.*

- `char e_i2c_start (void)`
  *Make the start bit.*

- `char e_i2c_restart (void)`
  *Make the restart bit.*

- `char e_i2c_ack (void)`
  *Make the acknowledgement bit.*

- `char e_i2c_nack (void)`
  *Make the non-acknowledgement bit.*

- `char e_i2c_read (char *buf)`
  *Read the I2C input register.*

- `char e_i2c_stop (void)`
  *Make the stop bit.*

- `char e_i2c_write (char byte)`
  *Write on the I2C output register.*

- `char e_i2c_enable (void)`
  *Enable special I2C interrupt.*

- `char e_i2c_disable (void)`
  *Disable special I2C interrupt.*

- `char e_i2c_reset (void)`
  *Reset the microcontroller for I2C uses.*
9.45.1 Detailed Description

Manage I2C basics.

Author:
   Code: Davis Daidie
   Doc: Jonathan Besuchet

9.45.2 Define Documentation

9.45.2.1 #define ACKNOWLEDGE 3
9.45.2.2 #define ERROR 10
9.45.2.3 #define OPERATION_OK 0
9.45.2.4 #define READ 4
9.45.2.5 #define RESTART 6
9.45.2.6 #define START 1
9.45.2.7 #define STOP 5
9.45.2.8 #define WRITE 2

9.45.3 Function Documentation

9.45.3.1 char e_i2c_ack (void)

Make the acknowledgement bit.

Returns:
   1 to confirm the operation and 0 for an error

9.45.3.2 char e_i2c_disable (void)

Disable special I2C interrupt.

Returns:
   1 to confirm the operation and 0 for an error

9.45.3.3 char e_i2c_enable (void)

Enable special I2C interrupt.

Returns:
   1 to confirm the operation and 0 for an error
9.45.3.4 char e_i2c_init (void)

Initialize the microcontroller for I2C uses.

**Returns:**
1 to confirm the operation and 0 for an error

9.45.3.5 char e_i2c_nack (void)

Make the non-acknowledgement bit.

**Returns:**
1 to confirm the operation and 0 for an error

9.45.3.6 char e_i2c_read (char *buf)

Read the I2C input register.

**Parameters:**
buf  A pointer to store the datas received

**Returns:**
1 to confirm the operation and 0 for an error

9.45.3.7 char e_i2c_reset (void)

Reset the microcontroller for I2C uses.

**Returns:**
1 to confirm the operation and 0 for an error

9.45.3.8 char e_i2c_restart (void)

Make the restart bit.

**Returns:**
1 to confirm the operation and 0 for an error

9.45.3.9 char e_i2c_start (void)

Make the start bit.

**Returns:**
1 to confirm the operation and 0 for an error
9.45.3.10 char e_i2c_stop (void)

Make the stop bit.

**Returns:**

1 to confirm the operation and 0 for an error

9.45.3.11 char e_i2c_write (char byte)

Write on the I2C output register.

**Parameters:**

`byte` What you want to send on I2C

**Returns:**

1 to confirm the operation and 0 for an error
### Functions

- **void e_i2cp_init (void)**  
  *Initialize the microcontroller for I2C uses.*

- **void e_i2cp_enable (void)**  
  *Enable special I2C interrupt.*

- **void e_i2cp_disable (void)**  
  *Disable special I2C interrupt.*

- **char e_i2cp_read (char device_add, char reg)**  
  *Read a specific register on a device.*

- **char e_i2cp_write (char device_add, char reg, char value)**  
  *Write a specific register on a device.*

### 9.46.1 Detailed Description

Manage I2C protocol.

This module manages the I2C protocol. The function's module is designed to directly send or receive data from or to a specified slave.

**Warning:**

This file must be included to communicate with an PO3030K camera through the I2C communication protocol.

**Author:**

Code: Davis Daidie  
Doc: Jonathan Besuchet

### 9.46.2 Function Documentation

#### 9.46.2.1 void e_i2cp_disable (void)

Disable special I2C interrupt.

**Returns:**

1 to confirm the operation and 0 for an error
9.46.2.2  void e_i2cp_enable (void)

Enable special I2C interrupt.

Returns:
   1 to confirm the operation and 0 for an error

9.46.2.3  void e_i2cp_init (void)

Initialize the microcontroller for I2C uses.

Returns:
   1 to confirm the operation and 0 for an error

9.46.2.4  char e_i2cp_read (char device_add, char reg)

Read a specific register on a device.

Parameters:
   device_add  The address of the device you want information
   reg  The register address you want read on the device

Returns:
   The readed value

9.46.2.5  char e_i2cp_write (char device_add, char reg, char value)

Write a specific register on a device.

Parameters:
   device_add  The address of the device you want information
   reg  The register address you want read on the device
   value  The data you want to write

Returns:
   1 to confirm the operation and 0 for an error
Manage I2C protocol.

#include "e_I2C_master_module.h"
#include "../motor_led/e_epuck_ports.h"

**Functions**

- void **e_i2cp_init** (void)
  
  *Initialize the microcontroller for I2C uses.*

- char **e_i2cp_write** (char device_add, char reg, char value)
  
  *Write a specific register on a device.*

- char **e_i2cp_read** (char device_add, char reg)
  
  *Read a specific register on a device.*

- char **e_i2cp_read_string** (char device_add, unsigned char read_buffer[], char start_address, char string_length)

- char **e_i2cp_write_string** (char device_add, unsigned char write_buffer[], char start_address, char string_length)

- void **e_i2cp_enable** (void)
  
  *Enable special I2C interrupt.*

- void **e_i2cp_disable** (void)
  
  *Disable special I2C interrupt.*

### 9.47.1 Detailed Description

Manage I2C protocol.

This module manages the I2C protocol. The function’s module are made to directly send or receive data from or to a specified slave.

**Warning:**

This file must be included to communicate with an PO3030K camera through the I2C communication protocol.

**Author:**

Code: Davis Daidie
Doc: Jonathan Besuchet

### 9.47.2 Function Documentation

#### 9.47.2.1 void **e_i2cp_disable** (void)

Disable special I2C interrupt.
Returns:

1 to confirm the operation and 0 for an error

9.47.2.2  void e_i2cp_enable (void)

Enable special I2C interrupt.

Returns:

1 to confirm the operation and 0 for an error

9.47.2.3  void e_i2cp_init (void)

Initialize the microcontroller for I2C uses.

Returns:

1 to confirm the operation and 0 for an error

9.47.2.4  char e_i2cp_read (char device_add, char reg)

Read a specific register on a device.

Parameters:

device_add  The address of the device you want information
reg  The register address you want read on the device

Returns:

The readed value

9.47.2.5  char e_i2cp_read_string (char device_add, unsigned char read_buffer[], char start_address, char string_length)

9.47.2.6  char e_i2cp_write (char device_add, char reg, char value)

Write a specific register on a device.

Parameters:

device_add  The address of the device you want information
reg  The register address you want read on the device
value  The data you want to write

Returns:

1 to confirm the operation and 0 for an error

9.47.2.7  char e_i2cp_write_string (char device_add, unsigned char write_buffer[], char start_address, char string_length)
Functions

- id EpuckPort()
- fclose(EpuckPort)

Variables

- clear EpuckPort

9.48.1 Function Documentation

9.48.1.1 id EpuckPort() [virtual]

9.48.1.2 fclose(EpuckPort)

9.48.2 Variable Documentation

9.48.2.1 catch could not open the delete the global variable clear EpuckPort clear global EpuckPort
9.49 matlab/matlab files/EpuckFlush.m File Reference

Functions

- id EpuckPort()
- flushinput (EpuckPort)
- flushoutput (EpuckPort)

Variables

- function[]
  - Matlab probably send old data if you don't do this flush

9.49.1 Function Documentation

9.49.1.1 id EpuckPort() [virtual]

  Type constraints

9.49.1.2 flushinput (EpuckPort)

9.49.1.3 flushoutput (EpuckPort)

9.49.2 Variable Documentation

9.49.2.1 Matlab probably send old data if you don't do this flush

9.49.2.2 function[]

Initial value:

EpuckFlush()
%EPUCKFLUSH Flush the input and output buffer of Matlab
% This function is useful each time you do a send / receive cycle with
% the e-puck. In fact
9.50 matlab/matlab files/EpuckGetData.m File Reference

Functions

- while \((i \sim 5)\) c
- switch i case if \((c \sim 'E')\) continue
- end case if \((c == 'I')\) i
- elseif \((c == 'C')\) i
- end end end if \((\text{receivedFormat} == 'I')\) size
- elseif \((\text{receivedFormat} == 'C')\) size

Variables

- function [data]
- \(i = 0\)
- continue
- \(\text{receivedFormat} = c\)
- \(\text{data} = \text{fread}(\text{EpuckPort}, \text{size}/2, 'int16')\)
- end return

9.50.1 Function Documentation

9.50.1.1 elseif \((\text{receivedFormat} = = 'C')\)

9.50.1.2 elseif \((c = = 'C')\)

9.50.1.3 end end end if \((\text{receivedFormat} = = 'I')\)

9.50.1.4 end case if \((c = = 'I')\)

9.50.1.5 end case if \((c \sim = 'E')\)

9.50.1.6 while \((i \sim 5)\)

9.50.2 Variable Documentation

9.50.2.1 continue

9.50.2.2 \(\text{data} = \text{fread}(\text{EpuckPort}, \text{size}/2, 'int16')\)

9.50.2.3 function[data]

Initial value:

\begin{verbatim}
EpuckGetData()
global EpuckPort
\end{verbatim}
9.50.2.4 else i = 0
9.50.2.5 receivedFormat = c
9.50.2.6 end return
9.51 matlab/matlab files/EpuckSendData.m File Reference

Functions

- if (strcmp(dataType,'char')) int8(data)
- fwrite (EpuckPort, size,'uint16')
- fwrite (EpuckPort, data,'int8')
- else int16 (data)
- fwrite (EpuckPort, 2 *size,'uint16')
- fwrite (EpuckPort, data,'int16')

Variables

- function [data]
- size = length(data)
- end return

9.51.1 Function Documentation

9.51.1.1 fwrite (EpuckPort, data, 'int16')

9.51.1.2 fwrite (EpuckPort, 2 * size, 'uint16')

9.51.1.3 fwrite (EpuckPort, data, 'int8')

9.51.1.4 fwrite (EpuckPort, size, 'uint16')

9.51.1.5 if (strcmp(dataType,'char'))

9.51.1.6 else int16 (data)

9.51.2 Variable Documentation

9.51.2.1 function[data]

Initial value:

EpuckSendData(data, dataType)
global EpuckPort

9.51.2.2 end return

9.51.2.3 size = length(data)
9.52 matlab/matlab files/OpenEpuck.m File Reference

Functions

- `port EpuckPort ()`
- `try fopen (EpuckPort)`

Variables

- `EpuckPort = serial(port,'BaudRate', 115200,'inputBuffersize',4096,'OutputBufferSize',4096,'ByteOrder','littleendian')`
- `catch could not open the port`
- `disp Error`
- `disp Could not open serial port return`

9.52.1 Function Documentation

9.52.1.1 `port EpuckPort ()` [virtual]

9.52.1.2 `try fopen (EpuckPort)`

9.52.2 Variable Documentation

9.52.2.1 `catch could not open the delete the global variable clear EpuckPort clear global EpuckPort = serial(port,'BaudRate',
115200,'inputBuffersize',4096,'OutputBufferSize',4096,'ByteOrder','littleendian')`

9.52.2.2 `disp Error`

9.52.2.3 `catch could not open the port`

9.52.2.4 `disp Could not open serial port return`
9.53 matlab/matlab.c File Reference

To communicate with matlab.
#include "matlab.h"
#include "./../motor_led/e_epuck_ports.h"
#include "./../uart/e_uart_char.h"

Functions

• void e_send_int_to_matlab (int *data, int array_size)
  The function to send int values to matlab.

• void e_send_char_to_matlab (char *data, int array_size)
  The function to send char values to matlab.

• int e_receive_int_from_matlab (int *data, int array_size)
  The function to receive int values from matlab.

• int e_receive_char_from_matlab (char *data, int array_size)
  The function to receive char values from matlab.

9.53.1 Detailed Description

To communicate with matlab.
This module manage the communication with matlab through bluetooth.

Author:
Code: Michael Bonani, Jonathan Besuchet, Doc: Jonathan Besuchet

9.53.2 Function Documentation

9.53.2.1 int e_receive_char_from_matlab (char * data, int array_size)

The function to receive char values from matlab.

Parameters:
  data    The array of char data you want to fill
  array_size  The length of the array

Returns:
  The number of char stored
9.53.2.2  int e_receive_int_from_matlab (int * data, int array_size)

The function to receive int values from matlab.

Parameters:

    data  The array of int data you want to fill
    array_size  The length of the array

Returns:

    The number of int stored

9.53.2.3  void e_send_char_to_matlab (char * data, int array_size)

The function to send char values to matlab.

Parameters:

    data  The array of char data you want to send
    array_size  The length of the array

9.53.2.4  void e_send_int_to_matlab (int * data, int array_size)

The function to send int values to matlab.

Parameters:

    data  The array of int data you want to send
    array_size  The length of the array
9.54 matlab/matlab.h File Reference

To communicate with matlab.

Functions

• void e_send_int_to_matlab (int *data, int array_size)
  The function to send int values to matlab.

• void e_send_char_to_matlab (char *data, int array_size)
  The function to send char values to matlab.

• int e_receive_int_from_matlab (int *data, int array_size)
  The function to receive int values from matlab.

• int e_receive_char_from_matlab (char *data, int array_size)
  The function to receive char values from matlab.

9.54.1 Detailed Description

To communicate with matlab.

This module manages the communication with matlab through Bluetooth.

Author:

Code: Michael Bonani, Jonathan Besuchet, Doc: Jonathan Besuchet

9.54.2 Function Documentation

9.54.2.1 int e_receive_char_from_matlab (char *data, int array_size)

The function to receive char values from matlab.

Parameters:

  *data* The array of char data you want to fill
  *array_size* The length of the array

Returns:

  The number of char stored

9.54.2.2 int e_receive_int_from_matlab (int *data, int array_size)

The function to receive int values from matlab.

Parameters:

  *data* The array of int data you want to fill
array_size  The length of the array

Returns:

The number of int stored

9.54.2.3  void e_send_char_to_matlab (char * data, int array_size)

The function to send char values to matlab.

Parameters:

data  The array of char data you want to send
array_size  The length of the array

9.54.2.4  void e_send_int_to_matlab (int * data, int array_size)

The function to send int values to matlab.

Parameters:

data  The array of int data you want to send
array_size  The length of the array
Manage the agendas (timer2).

```c
#include "e_agenda.h"
#include "../e_epuck_ports.h"
#include <stdlib.h>
```

**Defines**

- #define EXIT_OK 1

**Functions**

- void `e_start_agendas_processing` (void)
  
  Start the agendas processing.

- void `e_end_agendas_processing` (void)
  
  Stop all the agendas.

- int `e_activate_agenda` (void(*func)(void), int cycle)
  
  Activate an agenda.

- int `e_destroy_agenda` (void(*func)(void))
  
  Destroy an agenda.

- int `e_set_agenda_cycle` (void(*func)(void), int cycle)
  
  Change the cycle value of an agenda.

- int `e_reset_agenda` (void(*func)(void))
  
  Reset an agenda’s counter.

- int `e_pause_agenda` (void(*func)(void))
  
  Pause an agenda.

- int `e_restart_agenda` (void(*func)(void))
  
  Restart an agenda previously paused.

- void `__attribute__((interrupt, auto_psv))` ((interrupt, auto_psv))
  
  Interrupt from timer2.

**Variables**

- static Agenda * agenda_list = 0
9.55.1 Detailed Description

Manage the agendas (timer2).
This module manage the agendas with the timer2.
An agenda is a structure made to work as chained list. It contains: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.
Each times the timer2 has an interrupt, all the agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.
If one (or more) agenda has to be treated, his callback function is launch.

Author:
Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.55.2 Define Documentation

9.55.2.1 #define EXIT_OK 1

9.55.3 Function Documentation

9.55.3.1 void __attribute__((interrupt, auto_psv))

Interrupt from timer2.
Parse the chained list of agenda.
Increment counter only.
Check if agenda has to be treated according to the cycle value and current counter value.
Do it for number of cycle positive or null.
Check if a service has to be activated.

9.55.3.2 int e_activate_agenda (void(*)(void) func, int cycle)

Activate an agenda.
Activate an agenda and allocate memory for him if there isn’t already an agenda with the same callback function (the agenda is active but isn’t processed if he has a null cycle value).

Parameters:

func function called if the cycle value is reached by the counter
cycle cycle value in millisecond

Returns:

EXIT_OK (p. 314) if the agenda has been created, exit the programme otherwise
9.55.3.3 int e_destroy_agenda (void(*)(void) func)

Destroy an agenda.
Destroy the agenda with a given callback function.

Parameters:
func function to test

Returns:
EXIT_OK (p. 314) if the agenda has been destroyed, AG_NOT_FOUND (p. 319) otherwise

9.55.3.4 void e_end_agendas_processing (void)

Stop all the agendas.
Stop all the agendas by disabling Timer2

Warning:
the memory allocated for the agenda isn’t freed, use e_destroy_agenda (p. 320)(void (*)(func)(void)) for that.

See also:
e_destroy_agenda (p. 320)

9.55.3.5 int e_pause_agenda (void(*)(void) func)

Pause an agenda.
Pause an agenda but do not reset its information.

Parameters:
func function to pause

Returns:
EXIT_OK (p. 314) the agenda has been paused, AG_NOT_FOUND (p. 319) otherwise

9.55.3.6 int e_reset_agenda (void(*)(void) func)

Reset an agenda’s counter.
Reset an agenda’s counter with a given callback function.

Parameters:
func function to reset

Returns:
EXIT_OK (p. 314) if the cycle of the agenda has been reseted, AG_NOT_FOUND (p. 319) otherwise
Warning:

This function RESET the agenda, if you just want a pause tell e_pause_agenda (p. 321)(void (∗func)(void))

See also:

e_pause_agenda (p. 321)

9.55.3.7 int e_restart_agenda (void (∗)(void) func)

Restart an agenda previously paused.

Parameters:

func function to restart

Returns:

EXIT_OK (p. 314) if the agenda has been restarted, AG_NOT_FOUND (p. 319) otherwise

See also:

e_pause_agenda (p. 321)

9.55.3.8 int e_set_agenda_cycle (void (∗)(void) func, int cycle)

Change the cycle value of an agenda.

Parameters:

func function to test

cycle new cycle value in millisec/10

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been modified, AG_NOT_FOUND (p. 319) otherwise

9.55.3.9 void e_start_agendas_processing (void)

Start the agendas processing.

Start the agendas processing by starting the Timer2.

9.55.4 Variable Documentation

9.55.4.1 Agenda* agenda_list = 0 [static]

pointer on the end of agenda chained list
Manage the agendas (timer2).

Data Structures

- struct AgendaType
  - struct Agenda as chained list

Defines

- #define AG_ALREADY_CREATED 1
- #define AG_NOT_FOUND 2

Typedefs

- typedef struct AgendaType Agenda

Functions

- void e_start_agendas_processing (void)
  - Start the agendas processing.
- void e_end_agendas_processing (void)
  - Stop all the agendas.
- int e_activate_agenda (void(*)(void), int cycle)
  - Activate an agenda.
- int e_destroy_agenda (void(*)(void))
  - Destroy an agenda.
- int e_set_agenda_cycle (void(*)(void), int cycle)
  - Change the cycle value of an agenda.
- int e_reset_agenda (void(*)(void))
  - Reset an agenda’s counter.
- int e_pause_agenda (void(*)(void))
  - Pause an agenda.
- int e_restart_agenda (void(*)(void))
  - Restart an agenda previously paused.
9.56.1 Detailed Description

Manage the agendas (timer2).

This module manage the agendas with the timer2.

An agenda is a structure made to work as chained list. It contains: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each times the timer2 has an interrupt, all the agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, his callback function is launch.

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.56.2 Define Documentation

9.56.2.1 #define AG_ALREADY_CREATED 1

9.56.2.2 #define AG_NOT_FOUND 2

9.56.3 Typedef Documentation

9.56.3.1 typedef struct AgendaType Agenda

9.56.4 Function Documentation

9.56.4.1 int e_activate_agenda (void(*)(void) func, int cycle)

Activate an agenda.

Activate an agenda and allocate memory for him if there isn’t already an agenda with the same callback function (the agenda is active but isn’t processed if he has a null cycle value).

Parameters:

func function called if the cycle value is reached by the counter
cycle cycle value in millisec/10

Returns:

EXIT_OK (p. 314) if the agenda has been created, exit the programme otherwise

9.56.4.2 int e_destroy_agenda (void(*)(void) func)

Destroy an agenda.

Destroy the agenda with a given callback function.

Parameters:

func function to test
Returns:

- `EXIT_OK` (p. 314) if the agenda has been destroyed,
- `AG_NOT_FOUND` (p. 319) otherwise

Destroy the agenda with a given callback function

Parameters:

- `func` function to test

Returns:

- int return the success of the destruction (EXIT_OK for successfull, AG_NOT_FOUND for unsuccessfull).

### 9.56.4.3 `void e_end_agendas_processing (void)`

Stop all the agendas.

Stop all the agendas by disabling Timer2

Warning:

- the memory allocated for the agenda isn’t freed, use `e_destroy_agenda` (p. 320) for that.

See also:

- `e_destroy_agenda` (p. 320)

### 9.56.4.4 `int e_pause_agenda (void (*)(void) func)`

Pause an agenda.

Pause an agenda but do not reset its information.

Parameters:

- `func` function to pause

Returns:

- `EXIT_OK` (p. 314) the agenda has been paused,
- `AG_NOT_FOUND` (p. 319) otherwise

Pause an agenda but do not reset its information.

Parameters:

- `func` function to pause

### 9.56.4.5 `int e_reset_agenda (void (*)(void) func)`

Reset an agenda’s counter.

Reset an agenda’s counter with a given callback function.
Parameters:

\textit{func} function to reset

Returns:

\textbf{EXIT\_OK} (p. 314) if the cycle of the agenda has been reset, \textbf{AG\_NOT\_FOUND} (p. 319) otherwise

Warning:

This function \textbf{RESET} the agenda, if you just want a pause tell \texttt{e\_pause\_agenda} (p. 321)(\texttt{void (*func)(void)})

See also:

\texttt{e\_pause\_agenda} (p. 321)

Reset an agenda’s counter.
Reset an agenda’s counter with a given callback function

Parameters:

\textit{func} function to reset

9.56.4.6 \hspace{1em} \texttt{int e\_restart\_agenda \hspace{1em} (void(*)(void) func)}

Restart an agenda previously paused.
Restart an agenda previously paused.

Parameters:

\textit{func} function to restart

Returns:

\textbf{EXIT\_OK} (p. 314) if he agenda has been restarted, \textbf{AG\_NOT\_FOUND} (p. 319) otherwise

See also:

\texttt{e\_pause\_agenda} (p. 321)

Restart an agenda previously paused

Parameters:

\textit{func} function to restart

9.56.4.7 \hspace{1em} \texttt{int e\_set\_agenda\_cycle \hspace{1em} (void(*)(void) func, int cycle)}

Change the cycle value of an agenda.
Change the cycle value of an agenda with a given callback function.

Parameters:

\textit{func} function to test
cycle new cycle value in millisecond/10

Returns:

EXIT_OK (p. 314) if the cycle of the agenda has been modified, AG_NOT_FOUND (p. 319) otherwise

9.56.4.8 void e_start_agendas_processing (void)

Start the agendas processing.
Start the agendas processing by starting the Timer2.
Start the agendas processing.
Start the agendas processing by initialising the accounting structures.
Don’t activate any timer which is done by e_start_timer_processing.

See also:

e_start_timer_processing (p. 322)
#9.57  motor Led/Advance_One_Timer/eLed.c File Reference

Manage the LEDs with blinking possibility (timer2).

```c
#include "../e_epuck_ports.h"
#include "e_agenda.h"
```

**Defines**

- `#define LED_EFFECTS`  
  *For the preprocessor. Comment if you want not to use the LED effects.*

**Functions**

- `void e_set_led (unsigned int led_number, unsigned int value)`  
  *turn on/off the specified LED*

- `void e_led_clear (void)`  
  *turn off the 8 LEDs*

- `void e_set_body_led (unsigned int value)`  
  *turn on/off the body LED*

- `void e_set_front_led (unsigned int value)`  
  *turn on/off the front LED*

- `void e_blink_led (void)`  
  *Change the state of all LED.*

- `void e_blink_led0 (void)`  
  *Change the state of LED0.*

- `void e_blink_led1 (void)`  
  *Change the state of LED1.*

- `void e_blink_led2 (void)`  
  *Change the state of LED2.*

- `void e_blink_led3 (void)`  
  *Change the state of LED3.*

- `void e_blink_led4 (void)`  
  *Change the state of LED4.*

- `void e_blink_led5 (void)`  
  *Change the state of LED5.*

- `void e_blink_led6 (void)`
9.57 motor_led/advance_one_timer/e_led.c File Reference

Change the state of LED6.

- void e_blink_led7 (void)
  Change the state of LED7.

- void e_startLedBlinking (int cycle)
  Start blinking all LED.

- void e_stopLedBlinking (void)
  Stop blinking all LED.

- void e_setBlinkingCycle (int cycle)
  Change the blinking speed.

- void snakeLed (void)
  One led is on and turn clockwise.

- void flowLed (void)
  The leds go on from the front to the back and go off from the front to the back, etc.

- void k2000Led (void)
  The K2000 effect.

- void rightLed (void)
  The right LED are indicating the right side.

- void leftLed (void)
  The left LED are indicating the left side.

9.57.1 Detailed Description

Manage the LEDs with blinking possibility (timer2).

Here we use the agenda solution to make the LED blinking.

A little exemple for LEDs blinking with agenda (all LEDs blink with 100ms delay)

Warning:

this program uses the e_blink_led(void) (p. 270) function.

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
    e_init_port();
    e_activate_agenda(e_blink_led, 1000); // blink with 100ms
    e_start_agendas_processing();
    while(1) {}
}
```
See also:

*e_agenda.h* (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier, Jonathan Besuchet
Doc: Jonathan Besuchet

### 9.57.2 Define Documentation

#### 9.57.2.1 `#define LED_EFFECTS`

For the preprocessor. Comment if you want not to use the LED effects.

### 9.57.3 Function Documentation

#### 9.57.3.1 `void e_blink_led (void)`

Change the state of all LED.
Callback function for an agenda.

See also:

*AgendaType* (p. 57)

#### 9.57.3.2 `void e_blink_led0 (void)`

Change the state of LED0.
Callback function for an agenda.

See also:

*AgendaType* (p. 57)

#### 9.57.3.3 `void e_blink_led1 (void)`

Change the state of LED1.
Callback function for an agenda.

See also:

*AgendaType* (p. 57)

#### 9.57.3.4 `void e_blink_led2 (void)`

Change the state of LED2.
Callback function for an agenda.
See also:

AgendaType (p. 57)

9.57.3.5  void e_blink_led3 (void)

Change the state of LED3.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.6  void e_blink_led4 (void)

Change the state of LED4.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.7  void e_blink_led5 (void)

Change the state of LED5.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.8  void e_blink_led6 (void)

Change the state of LED6.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.57.3.9  void e_blink_led7 (void)

Change the state of LED7.
Callback function for an agenda.

See also:

AgendaType (p. 57)
9.57.3.10  void e_led_clear (void)

turn off the 8 LEDs
The e-puck has 8 green LEDs. This function turn all off.

Warning:

this function doesn’t turn off "body LED" and "front LED".

9.57.3.11  void e_set_blinking_cycle (int cycle)

Change the blinking speed.
This function use e_set_agenda_cycle (p. 322)(void (*func)(void), int cycle)

Parameters:

cycle  the number of cycle we wait before launching e_blink_led(void) (p. 270)"

See also:

e_blink_led (p. 270), e_set_agenda_cycle (p. 322)

9.57.3.12  void e_set_body_led (unsigned int value)

turn on/off the body LED
The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

value  0 (off), 1 (on) otherwise change the state

9.57.3.13  void e_set_front_led (unsigned int value)

turn on/off the front LED
The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value  0 (off), 1 (on) otherwise change the state

9.57.3.14  void e_set_led (unsigned int led_number, unsigned int value)

turn on/off the specified LED
The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number  between 0 and 7
value 0 (off), 1 (on) otherwise change the state

Warning:
if led_number is other than 0-7, all leds are set to the indicated value.

9.57.3.15  void e_start_led_blinking (int cycle)
Start blinking all LED.
Parameters:
cycle the number of cycle we wait before launching e_blink_led(void) (p. 270)
See also:
e_blink_led (p. 270), e_activate_agenda (p. 319)

9.57.3.16  void e_stop_led_blinking (void)
Stop blinking all LED.
This function use e_destroy_agenda (p. 320)(void (*func)(void))
See also:
e_destroy_agenda (p. 320)

9.57.3.17  void flow_led (void)
The leds go on from the front to the back and go off from the front to the back, etc.

9.57.3.18  void k2000_led (void)
The K2000 effect.

9.57.3.19  void left_led (void)
The left LED are indicating the left side.

9.57.3.20  void right_led (void)
The right LED are indicating the right side.

9.57.3.21  void snake_led (void)
One led is on and turn clockwise.
9.58 motor_led/advance_one_timer/fast_agenda/e_led.c File Reference

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

```c
#include "../../e_epuck_ports.h"
#include "e_agenda_fast.h"
```

**Functions**

- **void e_set_led**(unsigned int led_number, unsigned int value)
  
  turn on/off the specified LED

- **void e_led_clear**(void)
  
  turn off the 8 LEDs

- **void e_set_body_led**(unsigned int value)
  
  turn on/off the body LED

- **void e_set_front_led**(unsigned int value)
  
  turn on/off the front LED

- **void e_blink_led**(void)
  
  Change the state of all LED.

- **void e_blink_led0**(void)
  
  Change the state of LED0.

- **void e_blink_led1**(void)
  
  Change the state of LED1.

- **void e_blink_led2**(void)
  
  Change the state of LED2.

- **void e_blink_led3**(void)
  
  Change the state of LED3.

- **void e_blink_led4**(void)
  
  Change the state of LED4.

- **void e_blink_led5**(void)
  
  Change the state of LED5.

- **void e_blink_led6**(void)
  
  Change the state of LED6.

- **void e_blink_led7**(void)
  
  Change the state of LED7.
• void **e_start_led_blinking** (int cycle)
  
  *Start blinking all LED.*

• void **e_stop_led_blinking** (void)
  
  *Stop blinking all LED.*

• void **e_set_blinking_cycle** (int cycle)
  
  *Change the blinking speed.*

### 9.58.1 Detailed Description

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

Here we use the fast agenda solution to make the LED blinking.

See also:

**e_agenda.h** (p. 245)

Author:

- Code: Francesco Mondada
- Doc: Jonathan Besuchet

### 9.58.2 Function Documentation

#### 9.58.2.1 void e_blink_led (void)

Change the state of all LED.

Callback function for an agenda.

See also:

**AgendaType** (p. 57)

#### 9.58.2.2 void e_blink_led0 (void)

Change the state of LED0.

Callback function for an agenda.

See also:

**AgendaType** (p. 57)

#### 9.58.2.3 void e_blink_led1 (void)

Change the state of LED1.

Callback function for an agenda.
See also:
  AgendaType (p. 57)

9.58.2.4 void e_blink_led2 (void)
Change the state of LED2.
Callback function for an agenda.
See also:
  AgendaType (p. 57)

9.58.2.5 void e_blink_led3 (void)
Change the state of LED3.
Callback function for an agenda.
See also:
  AgendaType (p. 57)

9.58.2.6 void e_blink_led4 (void)
Change the state of LED4.
Callback function for an agenda.
See also:
  AgendaType (p. 57)

9.58.2.7 void e_blink_led5 (void)
Change the state of LED5.
Callback function for an agenda.
See also:
  AgendaType (p. 57)

9.58.2.8 void e_blink_led6 (void)
Change the state of LED6.
Callback function for an agenda.
See also:
  AgendaType (p. 57)
9.58.2.9 void e_blink_led7 (void)

Change the state of LED7.
Callback function for an agenda.

See also:
AgendaType (p. 57)

9.58.2.10 void e_led_clear (void)

turn off the 8 LEDs
The e-puck has 8 green LEDs. This function turn all off.

Warning:
this function doesn’t turn off "body LED” and "front LED”.

9.58.2.11 void e_set_blinking_cycle (int cycle)

Change the blinking speed.
This function use e_set_agenda_cycle(void (+func)(void), int cycle)

Parameters:
cycle the number of cycle we wait before launching "e_blink_led()"

See also:
e_blink_led (p. 270), e_set_agenda_cycle (p. 322)

9.58.2.12 void e_set_body_led (unsigned int value)

turn on/off the body LED
The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:
value 0 (off), 1 (on) otherwise change the state

9.58.2.13 void e_set_front_led (unsigned int value)

turn on/off the front LED
The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:
value 0 (off), 1 (on) otherwise change the state
9.58.2.14  void e_set_led (unsigned int led_number, unsigned int value)

turn on/off the specified LED

The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

  led_number  between 0 and 7
  value  0 (off), 1 (on) otherwise change the state

Warning:

  if led_number is other than 0-7, all LEDs are set to the indicated value.

9.58.2.15  void e_start_led_blinking (int cycle)

Start blinking all LED.

Parameters:

  cycle  the number of cycle we wait before launching "e_blink_led()"

See also:

  e_blink_led (p. 270), e_activate_agenda (p. 319)

9.58.2.16  void e_stop_led_blinking (void)

Stop blinking all LED.

This function use e_destroy_agenda(void (*func)(void))

See also:

  e_destroy_agenda (p. 320)
Manage the LEDs.
A little example for the LEDs (all the LEDs are blinking).

```c
#include "e_epuck_ports.h"
#include "e_led.h"
```

### Functions

- **void e_set_led (unsigned int led_number, unsigned int value)**
  
  *turn on/off the specified LED*

- **void e_set_body_led (unsigned int value)**
  
  *turn on/off the body LED*

- **void e_set_front_led (unsigned int value)**
  
  *turn on/off the front LED*

- **void e_led_clear (void)**
  
  *turn off the 8 LEDs*

### 9.59.1 Detailed Description

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

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_led.h>

int main(void)
{
    e_init_port();
    while(1)
    {
        long i;
        for(i=0; i<500000; i++)
            asm("NOP");
        e_set_led(8, 2); // switch the state of all leds
    }
}
```

**Author:**

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet
9.59.2 Function Documentation

9.59.2.1 void e_led_clear (void)

turn off the 8 LEDs
The e-puck has 8 green LEDs. This function turn all off.

Warning:
this function doesn’t turn off "body LED" and "front LED".

9.59.2.2 void e_set_body_led (unsigned int value)

turn on/off the body LED
The e-puck has a green LED that illuminate his body. With this function, you can change the state of this LED.

Parameters:
value 0 (off), 1 (on) otherwise change the state

9.59.2.3 void e_set_front_led (unsigned int value)

turn on/off the front LED
The e-puck has a red LED in the front. With this function, you can change the state of this LED.

Parameters:
value 0 (off), 1 (on) otherwise change the state

9.59.2.4 void e_set_led (unsigned int led_number, unsigned int value)

turn on/off the specified LED
The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:
led_number between 0 and 7
value 0 (off), 1 (on) otherwise change the state

Warning:
if led_number is other than 0-7, all leds are set to the indicated value.
9.60 motor_led/advance_one_timer/e_led.h File Reference

Manage the LEDs with blinking possibility (timer2).

Functions

- void **e_set_led** (unsigned int led_number, unsigned int value)
  - turn on/off the specified LED

- void **e_led_clear** (void)
  - turn off the 8 LEDs

- void **e_blinkLed** (void)
  - Change the state of all LED.

- void **e_blinkLed0** (void)
  - Change the state of LED0.

- void **e_blinkLed1** (void)
  - Change the state of LED1.

- void **e_blinkLed2** (void)
  - Change the state of LED2.

- void **e_blinkLed3** (void)
  - Change the state of LED3.

- void **e_blinkLed4** (void)
  - Change the state of LED4.

- void **e_blinkLed5** (void)
  - Change the state of LED5.

- void **e_blinkLed6** (void)
  - Change the state of LED6.

- void **e_blinkLed7** (void)
  - Change the state of LED7.

- void **e_set_bodyLed** (unsigned int value)
  - turn on/off the body LED

- void **e_set_frontLed** (unsigned int value)
  - turn on/off the front LED

- void **e_startLedBlinking** (int cycle)
  - Start blinking all LED.

- void **e_stopLedBlinking** (void)
Stop blinking all LED.

- void flow_led (void)
  
  The leds go on from the front to the back and go off from the front to the back, etc.

- void snakeLed (void)
  
  One led is on and turn clockwise.

- void k2000Led (void)
  
  The K2000 effect.

- void rightLed (void)
  
  The right LED are indicating the right side.

- void leftLed (void)
  
  The left LED are indicating the left side.

### 9.60.1 Detailed Description

Manage the LEDs with blinking possibility (timer2).

Here we use the agenda solution to make the LED blinking.

A little exemple for LEDs blinking with agenda (all LEDs blink with 100ms delay)

**Warning:**

this program uses the e_blink_led(void) (p. 270) function.

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_led.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
    e_init_port();
    e_activate_agenda(e_blink_led, 1000); //blink with 100ms
    e_start_agendas_processing();
    while(1) {}  
}
```

**See also:**

e_agenda.h (p. 245)

**Author:**

Code: Francesco Mondada, Lucas Meier, Jonathan Besuchet  
Doc: Jonathan Besuchet

### 9.60.2 Function Documentation

#### 9.60.2.1 void e_blink_led (void)

Change the state of all LED.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.2 void e_blink_led0 (void)

Change the state of LED0.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.3 void e_blink_led1 (void)

Change the state of LED1.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.4 void e_blink_led2 (void)

Change the state of LED2.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.5 void e_blink_led3 (void)

Change the state of LED3.
Callback function for an agenda.

See also:

AgendaType (p. 57)

9.60.2.6 void e_blink_led4 (void)

Change the state of LED4.
Callback function for an agenda.

See also:

AgendaType (p. 57)
9.60.2.7  void e_blink_led5 (void)

Change the state of LED5.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.60.2.8  void e_blink_led6 (void)

Change the state of LED6.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.60.2.9  void e_blink_led7 (void)

Change the state of LED7.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.60.2.10 void e_led_clear (void)

Turn off the 8 LEDs
The e-puck has 8 green LEDs. This function turn all off.

Warning:
   this function doesn’t turn off "body LED" and "front LED".

9.60.2.11 void e_set_body_led (unsigned int value)

Turn on/off the body LED
The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:
   value 0 (off), 1 (on) otherwise change the state

The e-puck has a green LED that illuminate his body. With this function, you can change the state of this LED.

Parameters:
   value 0 (off), 1 (on) otherwise change the state
9.60.2.12 void e_set_front_led (unsigned int value)

turn on/off the front LED
The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

The e-puck has a red LED in the front. With this function, you can change the state of this LED.

Parameters:

value 0 (off), 1 (on) otherwise change the state

9.60.2.13 void e_set_led (unsigned int led_number, unsigned int value)

turn on/off the specified LED
The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

led_number between 0 and 7
value 0 (off), 1 (on) otherwise change the state

Warning:
if led_number is other than 0-7, all leds are set to the indicated value.

9.60.2.14 void e_start_led_blinking (int cycle)

Start blinking all LED.

Parameters:

cycle the number of cycle we wait before launching e_blink_led(void) (p. 270)

See also:
e_blink_led (p. 270), e_activate_agenda (p. 319)

Parameters:

cycle the number of cycle we wait before launching "e_blink_led()"

See also:
e_blink_led (p. 270), e_activate_agenda (p. 319)
9.60.2.15  void e_stop_led_blinking (void)

Stop blinking all LED.

This function use e_destroy_agenda (p. 320)(void (*func)(void))

See also:

  e_destroy_agenda (p. 320)

This function use e_destroy_agenda(void (*func)(void))

See also:

  e_destroy_agenda (p. 320)

9.60.2.16  void flow_led (void)

The leds go on from the front to the back and go off from the front to the back, etc.

9.60.2.17  void k2000_led (void)

The K2000 effect.

9.60.2.18  void left_led (void)

The left LED are indicating the left side.

9.60.2.19  void right_led (void)

The right LED are indicating the right side.

9.60.2.20  void snake_led (void)

One led is on and turn clockwise.
9.61 motor_led/advance_one_timer/fast_agenda/e_led.h File Reference

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

Functions

- **void e_set_led** (unsigned int led_number, unsigned int value)
  
  **turn on/off the specified LED**

- **void e_led_clear** (void)
  
  **turn off the 8 LEDs**

- **void e_blink_led** (void)
  
  **Change the state of all LED.**

- **void e_blink_led0** (void)
  
  **Change the state of LED0.**

- **void e_blink_led1** (void)
  
  **Change the state of LED1.**

- **void e_blink_led2** (void)
  
  **Change the state of LED2.**

- **void e_blink_led3** (void)
  
  **Change the state of LED3.**

- **void e_blink_led4** (void)
  
  **Change the state of LED4.**

- **void e_blink_led5** (void)
  
  **Change the state of LED5.**

- **void e_blink_led6** (void)
  
  **Change the state of LED6.**

- **void e_blink_led7** (void)
  
  **Change the state of LED7.**

- **void e_set_body_led** (unsigned int value)
  
  **turn on/off the body LED**

- **void e_set_front_led** (unsigned int value)
  
  **turn on/off the front LED**

- **void e_start_led_blinking** (int cycle)
  
  **Start blinking all LED.**
• void e_stop_led_blinking (void)
  
  Stop blinking all LED.

9.61.1 Detailed Description

Manage the LEDs with blinking possibility (timer1, 2, 3).
Here we use the fast agenda solution to make the LED blinking.

See also:
  e_agenda.h (p. 245)

Author:
  
  Code: Francesco Mondada
  Doc: Jonathan Besuchet

9.61.2 Function Documentation

9.61.2.1 void e_blink_led (void)

Change the state of all LED.
Callback function for an agenda.

See also:
  AgendaType (p. 57)

9.61.2.2 void e_blink_led0 (void)

Change the state of LED0.
Callback function for an agenda.

See also:
  AgendaType (p. 57)

9.61.2.3 void e_blink_led1 (void)

Change the state of LED1.
Callback function for an agenda.

See also:
  AgendaType (p. 57)
9.61.2.4  void e_blink_led2 (void)

Change the state of LED2.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.61.2.5  void e_blink_led3 (void)

Change the state of LED3.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.61.2.6  void e_blink_led4 (void)

Change the state of LED4.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.61.2.7  void e_blink_led5 (void)

Change the state of LED5.
Callback function for an agenda.

See also:
   AgendaType (p. 57)

9.61.2.8  void e_blink_led6 (void)

Change the state of LED6.
Callback function for an agenda.

See also:
   AgendaType (p. 57)
9.61.2.9  void e_blink_led7 (void)

Change the state of LED7.
Callback function for an agenda.

See also:
  AgendaType (p. 57)

9.61.2.10  void e_led_clear (void)

turn off the 8 LEDs
The e-puck has 8 green LEDs. This function turn all off.

Warning:
  this function doesn’t turn off "body LED" and "front LED".

9.61.2.11  void e_set_body_led (unsigned int value)

turn on/off the body LED
The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:
  value 0 (off), 1 (on) otherwise change the state

9.61.2.12  void e_set_front_led (unsigned int value)

turn on/off the front LED
The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:
  value 0 (off), 1 (on) otherwise change the state
9.61.2.13 void e_set_led (unsigned int led_number, unsigned int value)

turn on/off the specified LED
The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

  led_number  between 0 and 7
  value  0 (off), 1 (on) otherwise change the state

Warning:

  if led_number is other than 0-7, all leds are set to the indicated value.

9.61.2.14 void e_start_led_blinking (int cycle)

Start blinking all LED.

Parameters:

  cycle  the number of cycle we wait before launching e_blink_led(void) (p. 270)

See also:

  e_blink_led (p. 270), e_activate_agenda (p. 319)

Parameters:

  cycle  the number of cycle we wait before launching "e_blink_led()"

See also:

  e_blink_led (p. 270), e_activate_agenda (p. 319)

9.61.2.15 void e_stop_led_blinking (void)

Stop blinking all LED.
This function use e_destroy_agenda (p. 320)(void (*func)(void))

See also:

  e_destroy_agenda (p. 320)

This function use e_destroy_agenda(void (*func)(void))

See also:

  e_destroy_agenda (p. 320)
9.62 motor_led/e_led.h File Reference

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

Functions

- void e_set_led (unsigned int led_number, unsigned int value)
  turn on/off the specified LED

- void e_led_clear (void)
  turn off the 8 LEDs

- void e_set_body_led (unsigned int value)
  turn on/off the body LED

- void e_set_front_led (unsigned int value)
  turn on/off the front LED

9.62.1 Detailed Description

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

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_led.h>

int main(void)
{
    e_init_port();
    while(1)
    {
        long i;
        for(i=0; i<500000; i++)
            asm("NOP");
        e_set_led(8, 2); // switch the state of all LEDs
    }
}
```

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.62.2 Function Documentation

9.62.2.1 void e_led_clear (void)

turn off the 8 LEDs
The e-puck has 8 green LEDs. This function turn all off.
Warning:

this function doesn’t turn off "body LED" and "front LED".

9.62.2.2  void e_set_body_led (unsigned int \texttt{value})

turn on/off the body LED
The e-puck has a green LED that illuminate his body. With this function, you can change the state of these LED.

Parameters:

\texttt{value} 0 (off), 1 (on) otherwise change the state

9.62.2.3  void e_set_front_led (unsigned int \texttt{value})

turn on/off the front LED
The e-puck has a red LED in the front. With this function, you can change the state of these LED.

Parameters:

\texttt{value} 0 (off), 1 (on) otherwise change the state

9.62.2.4  void e_set_led (unsigned int \texttt{led_number}, unsigned int \texttt{value})

turn on/off the specified LED
The e-puck has 8 green LEDs. With this function, you can change the state of these LEDs.

Parameters:

\texttt{led_number} between 0 and 7
\texttt{value} 0 (off), 1 (on) otherwise change the state

Warning:

if \texttt{led_number} is other than 0-7, all leds are set to the indicated value.
9.63  motor-led/advance_one_timer/e_motors.c File Reference

Manage the motors (with timer2).

```c
#include "./../e_epuck_ports.h"
#include "e_agenda.h"
#include <stdlib.h>
```

Defines

- `#define POWERSAVE`
- `#define TRESHV 650`
- `#define MAXV 601`

Functions

- `void run_left_motor (void)`
- `void run_right_motor (void)`
- `void e_init_motors (void)`
  
  *Initialize the motors’s agendas.*

- `void e_set_speed_left (int motor_speed)`
  
  *Manage the left motor speed.*

- `void e_set_speed_right (int motor_speed)`
  
  *Manage the right motor speed.*

- `void e_set_speed (int linear_speed, int angular_speed)`
  
  *Manage linear/angular speed.*

- `int e_get_steps_left ()`
  
  *Give the number of left motor steps.*

- `void e_set_steps_left (int set_steps)`
  
  *Set the number of left motor steps.*

- `int e_get_steps_right ()`
  
  *Give the number of right motor steps.*

- `void e_set_steps_right (int set_steps)`
  
  *Set the number of right motor steps.*

Variables

- `static int left_speed = 0`
- `static int right_speed = 0`
- `static int left_motor_phase = 0`
• static int right_motor_phase = 0
• static int nbr_steps_left = 0
• static int nbr_steps_right = 0

9.63.1 Detailed Description

Manage the motors (with timer2).

This module manage the motors with the agenda solution (timer2).

A little exemple to use the motors with agenda (e-puck turn on himself)

```c
#include <p3f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_motors.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
    e_init_port();
    e_init_motors();
    e_set_speed(-500, 500);
    e_start_agendas_processing();
    while(1) {}
}
```

See also:

   e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.63.2 Define Documentation

9.63.2.1 #define MAXV 601

9.63.2.2 #define POWERSAVE

9.63.2.3 #define TRESHV 650

9.63.3 Function Documentation

9.63.3.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

   The number of phases steps made since the left motor is running.
9.63.3.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.63.3.3 void e_init_motors (void)

Initialize the motors’s agendas.

This function initialize the agendas used by the motors. In fact it call e_activate_agenda (p. 319)(void (∗func)(void), int cycle) function.

See also:

  e_activate_agenda (p. 319)

9.63.3.4 void e_set_speed (int linear_speed, int angular_speed)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

  linear_speed  the speed in the axis of e-puck
  angular_speed  the rotation speed (trigonometric)

9.63.3.5 void e_set_speed_left (int motor_speed)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (∗func)(void), int cycle)).

Parameters:

  motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

  e_set_agenda_cycle (p. 322)

9.63.3.6 void e_set_speed_right (int motor_speed)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (∗func)(void), int cycle)).
Parameters:

*motor_speed* from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

*e_set_agenda_cycle* (p. 322)

9.63.3.7 void *e_set_steps_left* (int *set_steps*)

Set the number of left motor steps.

Parameters:

*set_steps* The number of changed phases that you want set.

9.63.3.8 void *e_set_steps_right* (int *set_steps*)

Set the number of right motor steps.

Parameters:

*set_steps* The number of changed phases that you want set.

9.63.3.9 void *run_left_motor* (void)

Change left motor phase according to the left_speed sign.

9.63.3.10 void *run_right_motor* (void)

Change right motor phase according to the right_speed sign

9.63.4 Variable Documentation

9.63.4.1 int *left_motor_phase* = 0 [static]

9.63.4.2 int *left_speed* = 0 [static]

9.63.4.3 int *nbr_steps_left* = 0 [static]

9.63.4.4 int *nbr_steps_right* = 0 [static]

9.63.4.5 int *right_motor_phase* = 0 [static]

9.63.4.6 int *right_speed* = 0 [static]
# e-motor API

## 9.64 e-motor LED/Advance one timer/Fast Agenda/e-motors.c File Reference

Manage the motors (with timer1, 2, 3).
```c
#include "../../e_epuck_ports.h"
#include "e_agenda_fast.h"
#include <stdlib.h>
```

### Defines
- `#define POWERSAVE`
- `#define TRESHV 650`
- `#define MAXV 601`

### Functions
- `void run_left_motor (void)`
- `void run_right_motor (void)`
- `void e_init_motors (void)`
  
  *Initialize the motors’ agendas.*
- `void e_set_speed_left (int motor_speed)`
  *Manage the left motor speed.*
- `void e_set_speed_right (int motor_speed)`
  *Manage the right motor speed.*
- `void e_set_speed (int linear_speed, int angular_speed)`
  *Manage linear/angular speed.*
- `int e_get_steps_left ()`
  *Give the number of left motor steps.*
- `void e_set_steps_left (int set_steps)`
  *Set the number of left motor steps.*
- `int e_get_steps_right ()`
  *Give the number of right motor steps.*
- `void e_set_steps_right (int set_steps)`
  *Set the number of right motor steps.*
Variables

- static int \texttt{left\_speed} = 0
- static int \texttt{right\_speed} = 0
- static int \texttt{left\_motor\_phase} = 0
- static int \texttt{right\_motor\_phase} = 0
- static int \texttt{nbr\_steps\_left} = 0
- static int \texttt{nbr\_steps\_right} = 0

\textbf{9.64.1 Detailed Description}

Manage the motors (with timer1, 2, 3). This module manage the motors with the fast agenda solution (timer1, 2, 3).

\textbf{See also:}

- \texttt{e\_agenda.h} (p. 245)

\textbf{Author:}

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

\textbf{9.64.2 Define Documentation}

- \texttt{#define MAXV 601}
- \texttt{#define POWERSAVE}
- \texttt{#define TRESHV 650}

\textbf{9.64.3 Function Documentation}

\textbf{9.64.3.1 int e_get_steps_left (void)}

Give the number of left motor steps.

\textbf{Returns:}

The number of phases steps made since the left motor is running.

\textbf{9.64.3.2 int e_get_steps_right (void)}

Give the number of right motor steps.

\textbf{Returns:}

The number of phases steps made since the right motor is running.
9.64.3.3  void e_init_motors (void)

Initialize the motors’s agendas.

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (*func)(void), int cycle)" function.

See also:
   e_activate_agenda (p. 319)

9.64.3.4  void e_set_speed (int linear_speed, int angular_speed)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:
   linear_speed  the speed in the axis of e-puck
   angular_speed the rotation speed (trigonometric)

9.64.3.5  void e_set_speed_left (int motor_speed)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (= speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (*func)(void), int cycle)").

Parameters:
   motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:
   e_set_agenda_cycle (p. 322)

9.64.3.6  void e_set_speed_right (int motor_speed)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (= speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (*func)(void), int cycle)").

Parameters:
   motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:
   e_set_agenda_cycle (p. 322)
9.64.3.7  void e_set_steps_left (int set_steps)

Set the number of left motor steps.

Parameters:

   
   set_steps  The number of changed phases that you want set.

9.64.3.8  void e_set_steps_right (int set_steps)

Set the number of right motor steps.

Parameters:

   
   set_steps  The number of changed phases that you want set.

9.64.3.9  void run_leftMotor (void)

Change left motor phase according to the left_speed signe.

9.64.3.10 void run_rightMotor (void)

Change right motor phase according to the right_speed signe

9.64.4  Variable Documentation

9.64.4.1  int left_motor_phase = 0  [static]

9.64.4.2  int left_speed = 0  [static]

9.64.4.3  int nbr_steps_left = 0  [static]

9.64.4.4  int nbr_steps_right = 0  [static]

9.64.4.5  int right_motor_phase = 0  [static]

9.64.4.6  int right_speed = 0  [static]
9.65 motor_led/e_motors.c File Reference

Manage the motors (with timer 4 and 5).
#include <stdlib.h>
#include "e_epuck_ports.h"
#include "e_init_port.h"
#include "e_motors.h"

Functions

• void __attribute__((interrupt, auto_psv, shadow))
  • void e_init_motors (void)
    Initialize the motors’s ports.

• void e_set_speed_left (int motor_speed)
  Manage the left speed.

• void e_set_speed_right (int motor_speed)
  Manage the right speed.

• int e_get_steps_left (void)
  Give the number of left motor steps.

• void e_set_steps_left (int set_steps)
  Set the number of left motor steps.

• int e_get_steps_right (void)
  Give the number of right motor steps.

• void e_set_steps_right (int set_steps)
  Set the number of right motor steps.

Variables

• static int left_speed = 0
• static int right_speed = 0
• static int nbr_pas_left = 0
• static int nbr_pas_right = 0

9.65.1 Detailed Description

Manage the motors (with timer 4 and 5).
This module manage the motors with two timers: timer4 (motor left) and timer5 (motor right).
Warning:
You can’t use this module to control the motors if you are using the camera, because the camera’s module also use timer4 and timer5.

A little exemple for the motors (e-puck turn on himself)

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_motors.h>

int main(void)
{
    e_init_motors();
    e_set_speed_left(500); //go forward on half speed
    e_set_speed_right(-500); //go backward on half speed
    while(1) {}
}
```

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

9.65.2 Function Documentation

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

9.65.2.2 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:
The number of phases steps made since the left motor is running.

9.65.2.3 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:
The number of phases steps made since the right motor is running.

9.65.2.4 void e_init_motors (void)

Initialize the motors’s ports.
Initialize the motors’s agendas.
This function initialize the ports used by the motors. In fact it call ”the e_init_port()” function.

See also:
e_init_port (p. 328)
9.65.2.5  void e_set_speed_left (int motor_speed)

Manage the left speed.
Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (\( \Rightarrow \) speed) is controled by the timer5.

Parameters:

  motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.65.2.6  void e_set_speed_right (int motor_speed)

Manage the right speed.
Manage the right motor speed.
This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (\( \Rightarrow \) speed) is controled by the timer4.

Parameters:

  motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.65.2.7  void e_set_steps_left (int set_steps)

Set the number of left motor steps.

Parameters:

  set_steps  The number of changed phases that you want set.

9.65.2.8  void e_set_steps_right (int set_steps)

Set the number of right motor steps.

Parameters:

  set_steps  The number of changed phases that you want set.

9.65.3  Variable Documentation

9.65.3.1  int left_speed = 0  [static]
9.65.3.2  int nbr_pas_left = 0  [static]
9.65.3.3  int nbr_pas_right = 0  [static]
9.65.3.4  int right_speed = 0  [static]
9.66 motor_led/advance_one_timer/e_motors.h File Reference

Manage the motors (with timer2).

Functions

- **void e_init_motors (void)**
  
  *Initialize the motors’s agendas.*

- **void e_set_speed_left (int motor_speed)**
  
  *Manage the left motor speed.*

- **void e_set_speed_right (int motor_speed)**
  
  *Manage the right motor speed.*

- **void e_set_speed (int linear_speed, int angular_speed)**
  
  *Manage linear/angular speed.*

- **void e_set_steps_left (int steps_left)**
  
  *Set the number of left motor steps.*

- **void e_set_steps_right (int steps_right)**
  
  *Set the number of right motor steps.*

- **int e_get_steps_left ()**
  
  *Give the number of left motor steps.*

- **int e_get_steps_right ()**
  
  *Give the number of right motor steps.*

9.66.1 Detailed Description

Manage the motors (with timer2).

This module manages the motors with the agenda solution (timer2).

A little example to use the motors with agenda (e-puck turn on himself)

```c
#include <p30f6014A.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_epuck_ports.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) {}
}
```

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
9.66.2 Function Documentation

9.66.2.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.66.2.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.66.2.3 void e_init_motors (void)

Initialize the motors’s agendas.

This function initialize the agendas used by the motors. In fact it call e_activate_agenda (p. 319)(void (*func)(void), int cycle) function.

See also:

e_activate_agenda (p. 319)

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (*func)(void), int cycle)" function.

See also:

e_activate_agenda (p. 319)

Initialize the motors’s agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)
Initialize the motors’s agendas.
This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)

9.66.2.4 void e_set_speed (int linear_speed, int angular_speed)

Manage linear/angular speed.
This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

linear_speed the speed in the axis of e-puck
angular_speed the rotation speed (trigonometric)

9.66.2.5 void e_set_speed_left (int motor_speed)

Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (∗func)(void), int cycle)).

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function "e_set_agenda_cycle( void (∗func)(void), int cycle")").

Parameters:

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the timer5.
Parameters:

`motor_speed` from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the left motor speed.

This function manages the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controlled by the timer3.

Parameters:

`motor_speed` from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.66.2.6 `void e_set_speed_right (int motor_speed)`

Manage the right motor speed.

This function manages the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controlled by the agenda (throw the function `e_set_agenda_cycle (p. 322)(void (*func)(void), int cycle)`).

Parameters:

`motor_speed` from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

`e_set_agenda_cycle (p. 322)`

This function manages the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controlled by the agenda (throw the function "e_set_agenda_cycle( void (*func)(void), int cycle)").

Parameters:

`motor_speed` from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

`e_set_agenda_cycle (p. 322)`

Manage the right motor speed.

This function manages the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controlled by the timer4.

Parameters:

`motor_speed` from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the right motor speed.

This function manages the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controlled by the timer3.
Parameters:

*motor_speed* from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.66.2.7 void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

*set_steps* The number of changed phases that you want set.

9.66.2.8 void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

*set_steps* The number of changed phases that you want set.
9.67 motor_led/advance_one_timer/fast_agenda/e_motors.h

File Reference

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

Functions

• void e_init_motors (void)
  
  Initialize the motors's agendas.

• void e_set_speed_left (int motor_speed)
  
  Manage the left motor speed.

• void e_set_speed_right (int motor_speed)
  
  Manage the right motor speed.

• void e_set_speed (int linear_speed, int angular_speed)
  
  Manage linear/angular speed.

• void e_set_steps_left (int steps_left)
  
  Set the number of left motor steps.

• void e_set_steps_right (int steps_right)
  
  Set the number of right motor steps.

• int e_get_steps_left ()
  
  Give the number of left motor steps.

• int e_get_steps_right ()
  
  Give the number of right motor steps.

9.67.1 Detailed Description

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

This module manage the motors with the fast agenda solution (timer1, 2, 3).

See also:

  e_agenda.h (p. 245)

Author:

  Code: Francesco Mondada, Lucas Meier
  Doc: Jonathan Besuchet
9.67.2 Function Documentation

9.67.2.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.67.2.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.67.2.3 void e_init_motors (void)

Initialize the motors's agendas.

This function initialize the agendas used by the motors. In fact it call e_activate_agenda (p. 319)(void (∗func)(void), int cycle) function.

See also:

   e_activate_agenda (p. 319)

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (∗func)(void), int cycle)" function.

See also:

   e_activate_agenda (p. 319)

Initialize the motors's agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

   e_init_port (p. 328)

Initialize the motors's agendas.

This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

   e_init_port (p. 328)
9.67.2.4  void e_set_speed (int linear_speed, int angular_speed)

Manage linear/angular speed.
This function manage the speed of the motors according to the desired linear and angular speed.

Parameters:

  * linear_speed  the speed in the axis of e-puck
  * angular_speed  the rotation speed (trigonometric)

9.67.2.5  void e_set_speed_left (int motor_speed)

Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (= speed) is controled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (∗func)(void), int cycle)).

Parameters:

  * motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

  * e_set_agenda_cycle (p. 322)

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (= speed) is controled by the agenda (throw the function "e_set_agenda_cycle(void (∗func)(void), int cycle)").

Parameters:

  * motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

  * e_set_agenda_cycle (p. 322)

Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (= speed) is controled by the timer5.

Parameters:

  * motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (= speed) is controled by the timer3.

Parameters:

  * motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.
9.67.2.6 \textbf{void} \textbf{e}\_set\_speed\_right (\textbf{int} \textbf{motor}\_speed)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (\(\Rightarrow\) speed) is controled by the agenda (throw the function \textbf{e}\_set\_agenda\_cycle (p. 322)(\textbf{void} (\textbf{\&func})(\textbf{void}), \textbf{int} \textbf{cycle})).

\textbf{Parameters:}

- \textbf{motor}\_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

\textbf{See also:}

- \textbf{e}\_set\_agenda\_cycle (p. 322)

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (\(\Rightarrow\) speed) is controled by the agenda (throw the function "e\_set\_agenda\_cycle(\textbf{void} (\textbf{\&func})(\textbf{void}), \textbf{int} \textbf{cycle}").

\textbf{Parameters:}

- \textbf{motor}\_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

\textbf{See also:}

- \textbf{e}\_set\_agenda\_cycle (p. 322)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (\(\Rightarrow\) speed) is controled by the timer4.

\textbf{Parameters:}

- \textbf{motor}\_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (\(\Rightarrow\) speed) is controled by the timer3.

\textbf{Parameters:}

- \textbf{motor}\_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.67.2.7 \textbf{void} \textbf{e}\_set\_steps\_left (\textbf{int} \textbf{set}\_steps)

Set the number of left motor steps.

\textbf{Parameters:}

- \textbf{set}\_steps The number of changed phases that you want set.
9.67.2.8  void e_set_steps_right (int set_steps)

Set the number of right motor steps.

**Parameters:**

- `set_steps`  The number of changed phases that you want set.
9.68 motorLed/e_motors.h File Reference

Manage the motors (with timer 4 and 5).

Functions

- **void e_init_motors (void)**
  
  Initialize the motors's agendas.

- **void e_set_speed_left (int motor_speed)**
  
  Manage the left motor speed.

- **void e_set_speed_right (int motor_speed)**
  
  Manage the right motor speed.

- **int e_get_steps_left (void)**
  
  Give the number of left motor steps.

- **int e_get_steps_right (void)**
  
  Give the number of right motor steps.

- **void e_set_steps_left (int set_steps)**
  
  Set the number of left motor steps.

- **void e_set_steps_right (int set_steps)**
  
  Set the number of right motor steps.

9.68.1 Detailed Description

Manage the motors (with timer 4 and 5).

This module manage the motors with two timers: timer4 (motor left) and timer5 (motor right).

Warning:

You can’t use this module to control the motors if you are using the camera, because the camera’s module also use timer4 and timer5.

A little exemple for the motors (e-puck turn on himself)

```c
#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) {} 
}
```
9.68.2 Function Documentation

9.68.2.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.68.2.2 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.68.2.3 void e_init_motors (void)

Initialize the motors’s agendas.

This function initialize the agendas used by the motors. In fact it call e_activate_agenda (p. 319)(void (+func)(void), int cycle) function.

See also:

  e_activate_agenda (p. 319)

This function initialize the agendas used by the motors. In fact it call "e_activate_agenda(void (+func)(void), int cycle)" function.

See also:

  e_activate_agenda (p. 319)

Initialize the motors’s agendas.

This function initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

  e_init_port (p. 328)

Initialize the motors’s agendas.

This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

  e_init_port (p. 328)
9.68.2.4  void e_set_speed_left (int motor_speed)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controlled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (*func)(void), int cycle)).

Parameters:

motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

e_set_agenda_cycle (p. 322)

9.68.2.5  void e_set_speed_right (int motor_speed)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controlled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (*func)(void), int cycle)).

Parameters:

motor_speed  from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.
Parameters:

\textit{motor\_speed} from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

\texttt{e\_set\_agenda\_cycle} (p. 322)

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (= speed) is controled by the agenda (throw the function "\texttt{e\_set\_agenda\_cycle(void (*func)(void), int cycle)}").

Parameters:

\textit{motor\_speed} from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

\texttt{e\_set\_agenda\_cycle} (p. 322)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (= speed) is controled by the timer4.

Parameters:

\textit{motor\_speed} from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (= speed) is controled by the timer3.

Parameters:

\textit{motor\_speed} from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.68.2.6 \hspace{1em} \textbf{void e\_set\_steps\_left (int set\_steps)}

Set the number of left motor steps.

Parameters:

\textit{set\_steps} The number of changed phases that you want set.

9.68.2.7 \hspace{1em} \textbf{void e\_set\_steps\_right (int set\_steps)}

Set the number of right motor steps.

Parameters:

\textit{set\_steps} The number of changed phases that you want set.
Manage the motors (with timer2).

```c
#include "./../e_epuck_ports.h"
#include "e_agenda.h"
#include <stdlib.h>
```

**Defines**

- `#define POWERSAVE`
- `#define TRESHV 650`
- `#define MAXV 601`

**Functions**

- `void run_left_motor (void)`
- `void run_right_motor (void)`
- `void e_init_motors (void)`
  
  Initialize the motors’s agendas.

- `void e_set_speed_left (int motor_speed)`
  
  Manage the left motor speed.

- `void e_set_speed_right (int motor_speed)`
  
  Manage the right motor speed.

- `void e_set_speed (int linear_speed, int angular_speed)`
  
  Manage linear/angular speed.

- `int e_get_steps_left ()`
  
  Give the number of left motor steps.

- `void e_set_steps_left (int set_steps)`
  
  Set the number of left motor steps.

- `int e_get_steps_right ()`
  
  Give the number of right motor steps.

- `void e_set_steps_right (int set_steps)`
  
  Set the number of right motor steps.

**Variables**

- `static int goal_active_left = 0`
- `static int goal_active_right = 0`
- `static int goal_steps_left = 0`
• static int goal_steps_right = 0
• static int left_speed = 0
• static int right_speed = 0
• static int left_motor_phase = 0
• static int right_motor_phase = 0
• static int nbr_steps_left = 0
• static int nbr_steps_right = 0

9.69.1 Detailed Description

Manage the motors (with timer2).

This module manage the motors with the agenda solution (timer2).

A little exemple to use the motors with agenda (e-puck turn on himself)

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_motors.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
    e_init_port();
    e_init_motors();
    e_set_speed(-500, 500);
    e_start_agendas_processing();
    while(1) {}  
}

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

9.69.2 Define Documentation

9.69.2.1 #define MAXV 601

9.69.2.2 #define POWERSAVE

9.69.2.3 #define TRESHV 650

9.69.3 Function Documentation

9.69.3.1 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.
9.69.3.2 int e_get_steps_right (void)

Give the number of right motor steps.

**Returns:**

The number of phases steps made since the right motor is running.

9.69.3.3 void e_init_motors (void)

Initialize the motors’s agendas.

This function initialize the agendas used by the motors. In fact it call e_activate_agenda (p. 319)(void (∗func)(void), int cycle) function.

**See also:**

e_activate_agenda (p. 319)

9.69.3.4 void e_set_speed (int linear_speed, int angular_speed)

Manage linear/angular speed.

This function manage the speed of the motors according to the desired linear and angular speed.

**Parameters:**

linear_speed the speed in the axis of e-puck

angular_speed the rotation speed (trigonometric)

9.69.3.5 void e_set_speed_left (int motor_speed)

Manage the left motor speed.

This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (∗func)(void), int cycle)).

**Parameters:**

motor_speed from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

**See also:**

e_set_agenda_cycle (p. 322)

9.69.3.6 void e_set_speed_right (int motor_speed)

Manage the right motor speed.

This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (=> speed) is controled by the agenda (throw the function e_set_agenda_cycle (p. 322)(void (∗func)(void), int cycle)).
Parameters:

*motor_speed* from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

See also:

* e_set_agenda_cycle (p. 322)

9.69.3.7  void e_set_steps_left (int *set_steps*)

Set the number of left motor steps.

Parameters:

*set_steps* The number of changed phases that you want set.

9.69.3.8  void e_set_steps_right (int *set_steps*)

Set the number of right motor steps.

Parameters:

*set_steps* The number of changed phases that you want set.

9.69.3.9  void run_left_motor (void)

Change left motor phase according to the left_speed sign.

9.69.3.10  void run_right_motor (void)

Change right motor phase according to the right_speed sign.
9.69.4 Variable Documentation

9.69.4.1 int goal_active_left = 0 [static]
9.69.4.2 int goal_active_right = 0 [static]
9.69.4.3 int goal_steps_left = 0 [static]
9.69.4.4 int goal_steps_right = 0 [static]
9.69.4.5 int left_motor_phase = 0 [static]
9.69.4.6 int left_speed = 0 [static]
9.69.4.7 int nbr_steps_left = 0 [static]
9.69.4.8 int nbr_steps_right = 0 [static]
9.69.4.9 int right_motor_phase = 0 [static]
9.69.4.10 int right_speed = 0 [static]
9.70 motorLed/advance_one_timer/e_remote_control.c File Reference

Manage the IR receiver module (timer2).

```c
#include "e_remote_control.h"
#include "../e_epuck_ports.h"
#include "e_agenda.h"
#include "e_led.h"
```

**Functions**

- **void e_init_remote_control (void)**
  
  Initialise the IR receiver ports.

- **void __attribute__((__interrupt__, auto_psv)) e_read_remote_control (void)**
  
  Read the signal and stock the information.

- **unsigned char e_get_check (void)**
  
  Read the check bit.

- **unsigned char e_get_address (void)**
  
  Read the adress of the commande.

- **unsigned char e_get_data (void)**
  
  Read the data of the command.

**Variables**

- static unsigned char `address_temp = 0`
- static unsigned char `data_temp = 0`
- static unsigned char `check_temp = 0`
- static unsigned char `address = 0`
- static unsigned char `data = 0`
- static unsigned char `check = 2`

**9.70.1 Detailed Description**

Manage the IR receiver module (timer2).

This module manage the IR receiver with the agenda solution (timer2).

A little exemple to manage the IR remote (the body LED change his state when you press a button of the IR controller).


```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_remote_control.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
    int ir_check;
    int previous_check = 0;
    e_init_port();
    e_init_remote_control();
    e_start_agendas_processing();
    while(1)
    {
        ir_check = e_get_check();
        if(ir_check != previous_check)
        {
            BODY_LED = BODY_LED^1;
            previous_check = ir_check;
        }
    }
}
```

See also:

e_agenda.h (p. 245)

Author:

Code: Francesco Mondada, Lucas Meier
Doc: Jonathan Besuchet

### 9.70.2 Function Documentation

#### 9.70.2.1 void __attribute__ ((__interrupt__, auto_psv))

#### 9.70.2.2 unsigned char e_get_address (void)

Read the adress of the commande.

**Returns:**

adress adress part of the signal

#### 9.70.2.3 unsigned char e_get_check (void)

Read the check bit.

**Returns:**

check check bit of the signal

#### 9.70.2.4 unsigned char e_get_data (void)

Read the data of the command.

**Returns:**

data data part of the signal

9.70.2.5 void e_init_remote_control (void)
Initialise the IR receiver ports.

9.70.2.6 void e_read_remote_control (void)
Read the signal and stock the information.

9.70.3 Variable Documentation

9.70.3.1 unsigned char address = 0 [static]
9.70.3.2 unsigned char address_temp = 0 [static]
9.70.3.3 unsigned char check = 2 [static]
9.70.3.4 unsigned char check_temp = 0 [static]
9.70.3.5 unsigned char data = 0 [static]
9.70.3.6 unsigned char data_temp = 0 [static]
9.71 motor_led/advance_one_timer/e_remote_control.h File Reference

Manage the LEDs with blinking possibility (timer2).

Defines

- #define BOTTOMR 10
- #define BOTTOML 11
- #define STANDBY 12
- #define MUTE 13
- #define VOL_UP 16
- #define VOL_DOWN 17
- #define CHAN_UP 32
- #define CHAN_DOWN 33
- #define I_II 35
- #define OUT_AUX_1 56

Functions

- void e_init_remote_control (void)
  
  Initialise the IR receiver ports.

- void e_read_remote_control (void)
  
  Read the signal and stock the information.

- unsigned char e_get_check (void)
  
  Read the check bit.

- unsigned char e_get_address (void)
  
  Read the adress of the commande.

- unsigned char e_get_data (void)
  
  Read the data of the command.

9.71.1 Detailed Description

Manage the LEDs with blinking possibility (timer2).

Manage the IR receiver module (timer2).

Here we use the agenda solution to make the LED blinking.

A little exemple for LEDs blinking with agenda (all LEDs blink with 100ms delay)

```c
#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) {} // infinite loop
}

See also:

e_agenda.h (p. 245)

Author:

Code: Valentin Longchamp
Doc: Jonathan Besuchet

This module manages the IR receiver with the agenda solution (timer2).

A little example to manage the IR remote (the body LED changes its state when you press a button of the IR controller).

```c
#include <p30f6014A.h>
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/advance_one_timer/e_remote_control.h>
#include <motor_led/advance_one_timer/e_agenda.h>

int main(void)
{
    int ir_check;
    int previous_check = 0;
    e_init_port();
    e_init_remote_control();
    e_start_agendas_processing();
    while(1)
    {
        ir_check = e_get_check();
        if(ir_check != previous_check)
        {
            BODY_LED = BODY_LED^1;
            previous_check = ir_check;
        }
    }
}
```

See also:

e_agenda.h (p. 245)

Author:

Jonathan Besuchet
9.71.2 Define Documentation

9.71.2.1 #define BOTTOML 11
9.71.2.2 #define BOTTOMR 10
9.71.2.3 #define CHAN_DOWN 33
9.71.2.4 #define CHAN_UP 32
9.71.2.5 #define I_II 35
9.71.2.6 #define MUTE 13
9.71.2.7 #define OUT_AUX_1 56
9.71.2.8 #define STANDBY 12
9.71.2.9 #define VOL_DOWN 17
9.71.2.10 #define VOL_UP 16

9.71.3 Function Documentation

9.71.3.1 unsigned char e_get_address (void)

Read the adress of the commande.

Returns:
  adress adress part of the signal

9.71.3.2 unsigned char e_get_check (void)

Read the check bit.

Returns:
  check check bit of the signal

9.71.3.3 unsigned char e_get_data (void)

Read the data of the command.

Returns:
  data data part of the signal

9.71.3.4 void e_init_remote_control (void)

Initialise the IR receiver ports.
9.71.3.5  void e_read_remote_control (void)

Read the signal and stock the information.
Manage the fast agendas (timer1, 2, 3).
#include "e_agenda_fast.h"
#include "../../e_epuck_ports.h"
#include <stdlib.h>

Defines

• #define EXIT_OK 1

Functions

• unsigned compute_gcd (unsigned u, unsigned v)
• unsigned my_ceil (float a)
• void e_set_timer_speed (int timer, unsigned speed)
• int search_best_fit (unsigned cycle)
• void recompute_speeds ()
• Agenda * find_function (void (*)(func)(void))
• unsigned assign_agenda (Agenda *agenda)
• void migrate (int timer)
• void e_start_agendas_processing (void)
  
  Initialise the accounting structure.

• void e_configure_timer (int timer)
  
  Configure the timer(s) used.

• void e_start_timer_processing (int timer)
  
  Start the timer(s) used.

• void e_end_agendas_processing (int timer)
  
  Stop an agenda running.

• void e_activate_agenda (void (*)(func)(void), unsigned cycle)
  
  Activate a fast agenda.

• void e_activate_motors (void (*)(func1)(void), void (*)(func2)(void))
• int e_set_motor_speed (void (*)(func)(void), unsigned cycle)
• int e_destroy_agenda (void (*)(func)(void))
  
  Destroy an agenda.

• int e_set_agenda_cycle (void (*)(func)(void), unsigned cycle)
• int e_reset_agenda (void (*)(func)(void))
  
  Reset an agenda.

• int e_pause_agenda (void (*)(func)(void))
Pause an agenda.

- int e_restart_agenda (void(*func)(void))
  
  Restart an agenda previously paused.

- void __attribute__((__interrupt__, auto_psv))
  
  Interrupt from timer1.

Variables

- static struct AgendaList agenda_list

9.72.1 Detailed Description

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

This module manage the fast agendas with the timer1, 2, 3.

An agenda is a structure made to work as chained list. It contains: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each times the timer1, 2, 3 has an interrupt, the corresponding agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, his callback function is launch.

Author:

  Code: Julien Hubert
  Doc: Jonathan Besuchet

9.72.2 Define Documentation

9.72.2.1 #define EXIT_OK 1

9.72.3 Function Documentation

9.72.3.1 void __attribute__((__interrupt__, auto_psv))

Interrupt from timer1.
Interrupt from timer3.
Interrupt from timer2.
Parse the chained list of agenda.
Increment counter only.
Check if agenda has to be treated according to the cycle value and current counter value.
Do it for number of cycle positive or null.
Check if a service has to be activated.
9.72.3.2 unsigned assign_agenda (Agenda * agenda)

9.72.3.3 unsigned compute_gcd (unsigned u, unsigned v)

9.72.3.4 void e_activate_agenda (void(*)(void) func, unsigned cycle)

Activate a fast agenda.
Activate an agenda and allocate memory for him if there isn’t already an agenda with the same callback function (the agenda is active but isn’t processed if he have a null cycle value).
The appropriate timer is automatically selected.

Parameters:

   func function called if the cycle value is reached by the counter
   cycle cycle value in milliseec/10

9.72.3.5 void e_activate_motors (void(*)(void) func1, void(*)(void) func2)

9.72.3.6 void e_configure_timer (int timer)

Configure the timer(s) used.
Configure one or all the timers to be used by the agenda

Parameters:

   timer the timer’s number to configure (between [0-3] with 0 configuring all of them)

9.72.3.7 int e_destroy_agenda (void(*)(void) func)

Destroy an agenda.
Destroy the agenda with a given callback function

Parameters:

   func function to test

Returns:

   int return the success of the destruction (EXIT_OK for successful, AG_NOT_FOUND for unsuccessful).

9.72.3.8 void e_end_agendas_processing (int timer)

Stop an agenda running.
Stop the agendas running on one particular timer (the memory allocated for the agenda isn’t freed, use e_destroy_agenda for that).

Parameters:

   timer the timer’s number [0-3]. 0 stops all the timers.
See also:

`e_destroy_agenda` (p. 320)

### 9.72.3.9 int e_pause_agenda (void(*)(void) func)

Pause an agenda.

Parameters:

- `func` function to pause

### 9.72.3.10 int e_reset_agenda (void(*)(void) func)

Reset an agenda.

Parameters:

- `func` function to reset

### 9.72.3.11 int e_restart_agenda (void(*)(void) func)

Restart an agenda previously paused.

Parameters:

- `func` function to restart

### 9.72.3.12 int e_set_agenda_cycle (void(*)(void) func, unsigned cycle)

### 9.72.3.13 int e_set_motor_speed (void(*)(void) func, unsigned cycle)

### 9.72.3.14 void e_set_timer_speed (int timer, unsigned speed)

### 9.72.3.15 void e_start_agendas_processing (void)

Initialise the accounting structure.

Start the agendas processing.

Don’t activate any timer which is done by `e_start_timer_processing`.

See also:

`e_start_timer_processing` (p. 322)
9.72.3.16  void e_start_timer_processing (int timer)

Start the timer(s) used.
Activate one or all the timers to be used by the agenda.

Parameters:

  timer  the timer’s number to activate (between [0-3] with 0 activating all of them)

9.72.3.17  Agenda* find_function (void(*)(void) func)
9.72.3.18  void migrate (int timer)
9.72.3.19  unsigned my_ceil (float a)
9.72.3.20  void recompute_speeds ()
9.72.3.21  int search_best_fit (unsigned cycle)

9.72.4  Variable Documentation

9.72.4.1  struct AgendaList agenda_list  [static]
9.73 motor_led/advance_one_timer/fast_agenda/e_agenda_fast.h

File Reference

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

Data Structures

- struct AgendaType
  
  Manage struct Agenda as chained list

- struct AgendaList
  
  Manage the different agendas lists.

Defines

- #define AG_ALREADY_CREATED 1
- #define AG_NOT_FOUND 2

Typedefs

- typedef struct AgendaType Agenda

Functions

- void e_start_agendas_processing (void)
  
  Start the agendas processing.

- void e_start_timer_processing (int timer)
  
  Start the timer(s) used.

- void e_end_agendas_processing (int timer)
  
  Stop an agenda running.

- void e_configure_timer (int timer)
  
  Configure the timer(s) used.

- void e_activate_agenda (void (*)(func)(void), unsigned cycle)
  
  Activate a fast agenda.

- void e_activate_motors (void (*)(func1)(void), void (*)(func2)(void))

- int e_set_motor_speed (void (*)(func)(void), unsigned cycle)

- int e_destroy_agenda (void (*)(func)(void))
  
  Destroy an agenda.

- int e_set_agenda_cycle (void (*)(func)(void), unsigned cycle)

- int e_reset_agenda (void (*)(func)(void))
  
  Reset an agenda’s counter.
• int e_pause_agenda (void(*)(void))(void))
  *Pause an agenda.*

• int e_restart_agenda (void(*)(void))(void))
  *Restart an agenda previously paused.*

9.73.1 Detailed Description

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

This module manages the fast agendas with the timer1, 2, 3.

An agenda is a structure made to work as a chained list. It contains: the function you want to launch, the time setup between two launching events, a counter to measure the current time, a pointer to the next element of the list.

Each time the timer1, 2, 3 has an interrupt, the corresponding agenda chained list is scanned to look if an agenda has to be treated according to the cycle value and current counter value.

If one (or more) agenda has to be treated, its callback function is launched.

Author:
  Code: Julien Hubert
  Doc: Jonathan Besuchet

9.73.2 Define Documentation

9.73.2.1 #define AG_ALREADY_CREATED 1

9.73.2.2 #define AG_NOT_FOUND 2

9.73.3 Typedef Documentation

9.73.3.1 typedef struct AgendaType Agenda

9.73.4 Function Documentation

9.73.4.1 void e_activate_agenda (void(*)(void) func, unsigned cycle)

Activate a fast agenda.

Activate an agenda and allocate memory for him if there isn’t already an agenda with the same callback function (the agenda is active but isn’t processed if he have a null cycle value).

The appropriate timer is automatically selected.

Parameters:

  func  function called if the cycle value is reached by the counter
  cycle cycle value in millisecond/10
9.73.4.2 void e_activate_motors (void (*)(void) func1, void (*)(void) func2)

9.73.4.3 void e_configure_timer (int timer)

Configure the timer(s) used.
Configure one or all the timers to be used by the agenda

Parameters:

   timer  the timer's number to configure (between [0-3] with 0 configuring all of them)

9.73.4.4 int e_destroy_agenda (void (*)(void) func)

Destroy an agenda.
Destroy the agenda with a given callback function.

Parameters:

   func  function to test

Returns:

   EXIT_OK (p. 314) if the agenda has been destroyed, AG_NOT_FOUND (p. 319) otherwise

Destroy the agenda with a given callback function

Parameters:

   func  function to test

Returns:

   int return the success of the destruction (EXIT_OK for successfull, AG_NOT_FOUND for unsuccess-
   full).

9.73.4.5 void e_end_agendas_processing (int timer)

Stop an agenda running.
Stop the agendas running on one particular timer (the memory allocated for the agenda isn’t freed, use
e_destroy_agenda for that).

Parameters:

   timer  the timer’s number [0-3]. 0 stops all the timers.

See also:

   e_destroy_agenda (p. 320)
9.73.4.6  int e_pause_agenda (void (*)(void) func)

Pause an agenda.
Pause an agenda but do not reset its information.

Parameters:
    func  function to pause

Returns:
    EXIT_OK (p. 314) the agenda has been paused, AG_NOT_FOUND (p. 319) otherwise

Warning:
This function RESET the agenda, if you just want a pause tell e_pause_agenda (p. 321)(void (*func)(void))

See also:
    e_pause_agenda (p. 321)

9.73.4.7  int e_reset_agenda (void (*)(void) func)

Reset an agenda’s counter.
Reset an agenda’s counter with a given callback function.

Parameters:
    func  function to reset

Returns:
    EXIT_OK (p. 314) if the cycle of the agenda has been reseted, AG_NOT_FOUND (p. 319) otherwise

9.73.4.8  int e_restart_agenda (void (*)(void) func)

Restart an agenda previously paused.
Restart an agenda previously paused.
Parameters:

\textit{func} function to restart

Returns:

\texttt{EXIT\_OK} (p. 314) if the agenda has been restarted, \texttt{AG\_NOT\_FOUND} (p. 319) otherwise

See also:

\texttt{e\_pause\_agenda} (p. 321)

Restart an agenda previously paused

Parameters:

\textit{func} function to restart

9.73.4.9 \texttt{int e\_set\_agenda\_cycle (void(*)(void) func, unsigned cycle)}

9.73.4.10 \texttt{int e\_set\_motor\_speed (void(*)(void) func, unsigned cycle)}

9.73.4.11 \texttt{void e\_start\_agendas\_processing (void)}

Start the agendas processing.
Start the agendas processing by starting the Timer2.
Start the agendas processing.
Start the agendas processing by initialising the accounting structures.
Don’t activate any timer which is done by \texttt{e\_start\_timer\_processing}.

See also:

\texttt{e\_start\_timer\_processing} (p. 322)

9.73.4.12 \texttt{void e\_start\_timer\_processing (int timer)}

Start the timer(s) used.
Activate one or all the timers to be used by the agenda.

Parameters:

\textit{timer} the timer’s number to activate (between [0-3] with 0 activating all of them)
motor_led/e_epuck_ports.h File Reference

Define all the useful names corresponding to e-puck's hardware.

```
#include "p30f6014A.h"
```

**Defines**

- `#define FOSC 7.3728e6`
- `#define PLL 8.0`
- `#define FCY ((FOSC*PLL)/(4.0))`
- `#define MILLISEC (FCY/1.0e3)`
- `#define MICROSEC (FCY/1.0e6)`
- `#define NANOSEC (FCY/1.0e9)`
- `#define TCY_PIC (1e9/FCY)`
- `#define INTERRUPT_DELAY (10*TCY_PIC)`
- `#define TRUE 1`
- `#define FALSE 0`
- `#define OUTPUT_PIN 0`
- `#define LED0 _LATA6`
- `#define LED1 _LATA7`
- `#define LED2 _LATA9`
- `#define LED3 _LATA12`
- `#define LED4 _LATA10`
- `#define LED5 _LATA13`
- `#define LED6 _LATA14`
- `#define LED7 _LATA15`
- `#define LED0_DIR _TRISA6`
- `#define LED1_DIR _TRISA7`
- `#define LED2_DIR _TRISA9`
- `#define LED3_DIR _TRISA12`
- `#define LED4_DIR _TRISA10`
- `#define LED5_DIR _TRISA13`
- `#define LED6_DIR _TRISA14`
- `#define LED7_DIR _TRISA15`
- `#define FRONT_LED _LATC1`
- `#define FRONT_LED_DIR _TRISC1`
- `#define BODY_LED _LATC2`
- `#define BODY_LED_DIR _TRISC2`
- `#define PULSE_IR0 _LATF7`
- `#define PULSE_IR1 _LATF8`
- `#define PULSE_IR2 _LATG0`
- `#define PULSE_IR3 _LATG1`
- `#define PULSE_IR0_DIR _TRISF7`
- `#define PULSE_IR1_DIR _TRISF8`
- `#define PULSE_IR2_DIR _TRISG0`
- `#define PULSE_IR3_DIR _TRISG1`
- `#define IR0 8`
- `#define IR1 9`
- `#define IR2 10`
• #define IR3 11
• #define IR4 12
• #define IR5 13
• #define IR6 14
• #define IR7 15
• #define MIC1 2
• #define MIC2 3
• #define MIC3 4
• #define ACCX 5
• #define ACCY 6
• #define ACCZ 7
• #define AUDIO_ON _LATF0
• #define AUDIO_ON_DIR _TRISF0
• #define MOTOR1_PHA _LATD0
• #define MOTOR1_PHB _LATD1
• #define MOTOR1_PHC _LATD2
• #define MOTOR1_PHD _LATD3
• #define MOTOR2_PHA _LATD4
• #define MOTOR2_PHB _LATD5
• #define MOTOR2_PHC _LATD6
• #define MOTOR2_PHD _LATD7
• #define MOTOR1_PHA_DIR _TRISD0
• #define MOTOR1_PHB_DIR _TRISD1
• #define MOTOR1_PHC_DIR _TRISD2
• #define MOTOR1_PHD_DIR _TRISD3
• #define MOTOR2_PHA_DIR _TRISD4
• #define MOTOR2_PHB_DIR _TRISD5
• #define MOTOR2_PHC_DIR _TRISD6
• #define MOTOR2_PHD_DIR _TRISD7
• #define CAM_RESET _LATC13
• #define CAM_RESET_DIR _TRISC13
• #define SIO_D _LATG3
• #define SIO_D_DIR _TRISG3
• #define SIO_C _LATG2
• #define SIO_C_DIR _TRISG2
• #define INPUT_PIN 1
• #define BATT_LOW _RF1
• #define BATT_LOW_DIR _TRISF1
• #define SELECTOR0 _RG6
• #define SELECTOR1 _RG7
• #define SELECTOR2 _RG8
• #define SELECTOR3 _RG9
• #define SELECTOR0_DIR _TRISG6
• #define SELECTOR1_DIR _TRISG7
• #define SELECTOR2_DIR _TRISG8
• #define SELECTOR3_DIR _TRISG9
• #define REMOTE _RF6
• #define REMOTE_DIR _TRISF6
• #define CAM_DATA PORTD;
• #define CAM_y0 _RD8
9.74.1 Detailed Description

Define all the useful names corresponding of e-puck’s hardware.

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet
9.74.2 Define Documentation

9.74.2.1 #define ACCX 5
9.74.2.2 #define ACCY 6
9.74.2.3 #define ACCZ 7
9.74.2.4 #define AUDIO_ON _LATF0
9.74.2.5 #define AUDIO_ON_DIR _TRISF0
9.74.2.6 #define BATT_LOW _RF1
9.74.2.7 #define BATT_LOW_DIR _TRISF1
9.74.2.8 #define BODY_LED _LATC2
9.74.2.9 #define BODY_LED_DIR _TRISC2
9.74.2.10 #define CAM_DATA PORTD;
9.74.2.11 #define CAM_HREF _RC3
9.74.2.12 #define CAM_HREF_DIR _TRISC3
9.74.2.13 #define CAM_PCLK _RC14
9.74.2.14 #define CAM_PCLK_DIR _TRISC14
9.74.2.15 #define CAM_PWDN _RC2
9.74.2.16 #define CAM_PWDN_DIR _TRISC2
9.74.2.17 #define CAM_RESET _LATC13
9.74.2.18 #define CAM_RESET_DIR _TRISC13
9.74.2.19 #define CAM_VSYNC _RC4
9.74.2.20 #define CAM_VSYNC_DIR _TRISC4
9.74.2.21 #define CAM_y0 _RD8
9.74.2.22 #define CAM_y0_DIR _TRISD8
9.74.2.23 #define CAM_y1 _RD9
9.74.2.24 #define CAM_y1_DIR _TRISD9
9.74.2.25 #define CAM_y2 _RD10
9.74.2.26 #define CAM_y2_DIR _TRISD10
9.74.2.27 #define CAM_y3 _RD11
9.74.2.28 #define CAM_y3_DIR _TRISD11
9.74.2.29 #define CAM_y4 _RD12
9.74.2.30 #define CAM_y4_DIR _TRISD12
9.75 motor_led/e_init_port.c File Reference

Initialize the ports on standard configuration.
#include "e_epuck_ports.h"

Functions

- _FOSC (CSW_FSCM_OFF & XT_PLL8)
- _FWDT (WDT_OFF)
- _FBORPOR (PBOR_OFF & MCLR_EN)
- _FGS (CODE_PROT_OFF)
- void e_init_port (void)

  Initialize all ports (in/out).

9.75.1 Detailed Description

Initialize the ports on standard configuration.

Author:
Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

9.75.2 Function Documentation

9.75.2.1 _FBORPOR (PBOR_OFF & MCLR_EN)

9.75.2.2 _FGS (CODE_PROT_OFF)

9.75.2.3 _FOSC (CSW_FSCM_OFF & XT_PLL8)

9.75.2.4 _FWDT (WDT_OFF)

9.75.2.5 void e_init_port (void)

Initialize all ports (in/out).

Call this method to set all the standards output components (LEDs, IR, camera, motors, I2C, audio) on their defaults values and set their corresponding PIN to "output". The method also set the corresponding PIN to "input" for all the standards inputs components (IR receiver, selector, camera, battery level).
9.76 motor_led/e_init_port.h File Reference

Initialize the ports on standard configuration.

Functions

- void e_init_port (void)
  
  Initialize all ports (in/out).

9.76.1 Detailed Description

Initialize the ports on standard configuration.

Author:

  Code: Michael Bonani, Francesco Mondada, Davis Dadie
  Doc: Jonathan Besuchet

9.76.2 Function Documentation

9.76.2.1 void e_init_port (void)

Initialize all ports (in/out).

Call this method to set all the standards output components (LEDs, IR, camera, motors, I2C, audio) on their defaults values and set their corresponding PIN to "output". The method also set the corresponding PIN to "input" for all the standards inputs components (IR receiver, selector, camera, battery level).
Initialize the ports on standard configuration.

```c
#include <stdlib.h>
#include "e_epuck_ports.h"
#include "e_init_port.h"
```

### Functions

- **void** `__attribute__((interrupt, auto_psv, shadow))`
- **int** `e_get_steps_left (void)`
  
  *Give the number of left motor steps.*

- **void** `e_set_steps_left (int set_steps)`
  
  *Set the number of left motor steps.*

- **int** `e_get_steps_right (void)`
  
  *Give the number of right motor steps.*

- **void** `e_set_steps_right (int set_steps)`
  
  *Set the number of right motor steps.*

- **void** `e_set_speed_left (int motor_speed)`
  
  *Manage the left speed.*

- **void** `e_set_speed_right (int motor_speed)`
  
  *Manage the right speed.*

- **void** `e_init_motors (void)`
  
  *Initialize the motors's ports.*

### Variables

- **static int** `left_speed = 0`
- **static int** `right_speed = 0`
- **static int** `nbr_pas_left = 0`
- **static int** `nbr_pas_right = 0`
- **static int** `motor_counter_left = 0`
- **static int** `motor_counter_right = 0`
- **static int** `motor_counter_left_init = 0`
- **static int** `motor_counter_right_init = 0`
9.77.1 Detailed Description

Initialize the ports on standard configuration.
Manage the motors (with timer3).

Author:

Code: Michael Bonani, Francesco Mondada, Davis Dadie
Doc: Jonathan Besuchet

This module manage the two motors with one timer: timer3.

Author:

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

9.77.2 Function Documentation

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

9.77.2.2 int e_get_steps_left (void)

Give the number of left motor steps.

Returns:

The number of phases steps made since the left motor is running.

9.77.2.3 int e_get_steps_right (void)

Give the number of right motor steps.

Returns:

The number of phases steps made since the right motor is running.

9.77.2.4 void e_init_motors (void)

Initialize the motors’s ports.
Initialize the motors’s agendas.
This function configure timer3 and initialize the ports used by the motors. In fact it call "the e_init_port()" function.

See also:

e_init_port (p. 328)
9.77.2.5  void e_set_speed_left (int motor_speed)

Manage the left speed.
Manage the left motor speed.
This function manage the left motor speed by changing the MOTOR1 phases. The changing phases frequency (\( \Rightarrow \) speed) is controled by the timer3.

Parameters:

\( \text{motor\_speed} \) from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.77.2.6  void e_set_speed_right (int motor_speed)

Manage the right speed.
Manage the right motor speed.
This function manage the right motor speed by changing the MOTOR2 phases. The changing phases frequency (\( \Rightarrow \) speed) is controled by the timer3.

Parameters:

\( \text{motor\_speed} \) from -1000 to 1000 give the motor speed in steps/s, positive value to go forward and negative to go backward.

9.77.2.7  void e_set_steps_left (int set_steps)

Set the number of left motor steps.

Parameters:

\( \text{set\_steps} \) The number of changed phases that you want set.

9.77.2.8  void e_set_steps_right (int set_steps)

Set the number of right motor steps.

Parameters:

\( \text{set\_steps} \) The number of changed phases that you want set.
9.77.3 Variable Documentation

9.77.3.1 int left_speed = 0 [static]
9.77.3.2 int motor_counter_left = 0 [static]
9.77.3.3 int motor_counter_left_init = 0 [static]
9.77.3.4 int motor_counter_right = 0 [static]
9.77.3.5 int motor_counter_right_init = 0 [static]
9.77.3.6 int nbr_pas_left = 0 [static]
9.77.3.7 int nbr_pas_right = 0 [static]
9.77.3.8 int right_speed = 0 [static]
9.79  uart/e_uart_char.h File Reference

Manage UART.

Functions

- **void** e_init_uart1 (void)
  
  *Init uart 1 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.*

- **int** e_ischar_uart1 ()
  
  *Check if something is comming on uart 1.*

- **int** e_getchar_uart1 (char *car)
  
  *If available, read 1 char and put it in pointer.*

- **void** e_send_uart1_char (const char *buff, int length)
  
  *Send a buffer of char of size length.*

- **int** e_uart1_sending (void)
  
  *To check if the sending operation is done.*

- **void** e_init_uart2 (void)
  
  *Init uart 2 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.*

- **int** e_ischar_uart2 ()
  
  *Check if something is comming on uart 2.*

- **int** e_getchar_uart2 (char *car)
  
  *If available, read 1 char and put it in pointer.*

- **void** e_send_uart2_char (const char *buff, int length)
  
  *Send a buffer of char of size length.*

- **int** e_uart2_sending (void)
  
  *To check if the sending operation is done.*

Variables

- **void** *e_uart1_int_clr_addr
  
- **int** e_uart1_int_clr_mask

- **void** *e_uart2_int_clr_addr
  
- **int** e_uart2_int_clr_mask
9.79.1 Detailed Description

Manage UART.

This module manage all the UART ressource.

The e-puck’s microcontroller has two integrated UART: UART1 and UART2.

A little exemple to communicate 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.

```c
#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 coming from the user
        while(e_uart1_sending() || !e_ischar_uart1()) {}
        e_getchar_uart1(&car); // read the character entered...
        e_send_uart1_char("You have wrote: ", 16);
        e_send_uart1_char(&car, 1); //... and resend him to uart.
        e_send_uart1_char("\r\n\r\n", 4);
    }
}
```

Author:

Code: Michael Bonani
Doc: Jonathan Besuchet

9.79.2 Function Documentation

9.79.2.1 int e_getchar_uart1 (char * car)

If available, read 1 char and put it in pointer.

Parameters:

`car` The pointer where the character will be stored if available

Returns:

1 if a char has been readed, 0 if no char is available

9.79.2.2 int e_getchar_uart2 (char * car)

If available, read 1 char and put it in pointer.
Parameters:

- car The pointer where the character will be stored if available

Returns:

- 1 if a char has been readed, 0 if no char is available

9.79.2.3 void e_init_uart1 (void)

Init uart 1 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.

9.79.2.4 void e_init_uart2 (void)

Init uart 2 at 115200bps, 8 data bits, 1 stop bit, Enable ISR for RX and TX.

9.79.2.5 int e_ischar_uart1 ()

Check if something is coming on uart 1.

Returns:

- the number of characters available, 0 if none are available

9.79.2.6 int e_ischar_uart2 ()

Check if something is coming on uart 2.

Returns:

- the number of characters available, 0 if none are available

9.79.2.7 void e_send_uart1_char (const char * buff, int length)

Send a buffer of char of size length.

Parameters:

- buff The top of the array where the data are stored
- length The length of the array to send

9.79.2.8 void e_send_uart2_char (const char * buff, int length)

Send a buffer of char of size length.

Parameters:

- buff The top of the array where the datas are stored
- length The length of the array
9.79.2.9  int e_uart1_sending (void)

To check if the sending operation is done.

Returns:
   1 if buffer sending is in progress, return 0 if not

9.79.2.10 int e_uart2_sending (void)

To check if the sending operation is done.

Returns:
   1 if buffer sending is in progress, return 0 if not

9.79.3  Variable Documentation

9.79.3.1  void* e_uart1_int_clr_addr
9.79.3.2  int e_uart1_int_clr_mask
9.79.3.3  void* e_uart2_int_clr_addr
9.79.3.4  int e_uart2_int_clr_mask
Index

_FBORPOR
  e_init_port.c, 327
_FGS
  e_init_port.c, 327
_FOSC
  e_init_port.c, 327
_FWDT
  e_init_port.c, 327
__attribute__
  advance_ad_scan/e_ad_conv.c, 75
e_agenda.c, 242
e_agenda_fast.c, 314
e_fft.c, 214
e_I2C_master_module.c, 220
e_input_signal.c, 217
e_motors.c, 285
e_motors_timer3.c, 330
e_prox.c, 94
e_prox_timer2.c, 107
e_remote_control.c, 307
e_twiddle_factors.c, 218
fast_2_timer/e_timers.c, 189
slow_3_timer/e_timers.c, 192
_po3030k_buffer
  fast_2_timer/e_timers.c, 191
_po3030k_current_row
  fast_2_timer/e_timers.c, 191
_po3030k_img_ready
  fast_2_timer/e_timers.c, 191
_po3030k_line_conf
  fast_2_timer/e_timers.c, 191
_po3030k_row
  fast_2_timer/e_timers.c, 191

a_d/ Directory Reference, 37
a_d/advance_ad_scan/ Directory Reference, 38
a_d/advance_ad_scan/e_acc.c, 63
a_d/advance_ad_scan/e_acc.h, 69
a_d/advance_ad_scan/e_ad_conv.c, 74
a_d/advance_ad_scan/e_ad_conv.h, 78
a_d/advance_ad_scan/e_micro.c, 82
a_d/advance_ad_scan/e_micro.h, 86
a_d/advance_ad_scan/e_prox.c, 90
a_d/advance_ad_scan/e_prox.h, 96
a_d/e_accelerometer.c, 102
a_d/e_ad_conv.c, 77
a_d/e_ad_conv.h, 81
a_d/e_micro.c, 84
a_d/e_micro.h, 88
a_d/e_prox.c, 93
a_d/e_prox.h, 99
a_d/e_prox_timer2.c, 106
ACC_PROX_PERIOD
  advance_ad_scan/e_ad_conv.h, 79
ACC_PROX_SAMP_FREQ
  advance_ad_scan/e_ad_conv.h, 79
ACC_SAMP_NB
  advance_ad_scan/e_ad_conv.h, 79
acc_x
  e_acc.c, 68
    TypeAccRaw, 61
acc_y
  e_acc.c, 68
    TypeAccRaw, 61
acc_z
  e_acc.c, 68
    TypeAccRaw, 61
acceleration
  e_acc.c, 68
    TypeAccSpheric, 62
ACCX
  e_epuck_ports.h, 326
ACCX_BUFFER
  e_acc.h, 71
ACCY
  e_epuck_ports.h, 326
ACCY_BUFFER
  e_acc.h, 71
ACCZ
  e_epuck_ports.h, 326
ACCZ_BUFFER
  e_acc.h, 71
ACKNOWLEDGE
  e_I2C_master_module.h, 224
activate
  AgendaType, 57
ADCOFF_BASE
  fast_2_timer/e_registers.c, 160
    slow_3_timer/e_registers.c, 176
INDEX

ADCS_3_CHAN
advance_ad_scan/e_ad_conv.h, 79

ADCS_5_CHAN
advance_ad_scan/e_ad_conv.h, 79

ADCS_6_CHAN
advance_ad_scan/e_ad_conv.h, 79

address
BtDevice, 59
BtEPuck, 60
e_remote_control.c, 308

address_temp
e_remote_control.c, 308

ADDRHDATA_IN_PACKET_OFFSET
ComModule.c, 211

ADDRLDATA_IN_PACKET_OFFSET
ComModule.c, 211

advance_ad_scan/e_ad_conv.c
__attribute__, 75
e_acc_scan, 76
e_ad_is_acquisition_completed, 75
e_ad_is_array_filled, 75
e_ad_scan_off, 75
e_ad_scan_on, 75
e_ambient_and_reflected_ir, 76
e_ambient_ir, 76
e_init_ad_scan, 75
e_last_acc_scan_id, 76
e_last_mic_scan_id, 76
e_mic_scan, 76
is_ad_acquisition_completed, 76
is_ad_array_filled, 76
micro_only, 76

advance_ad_scan/e_ad_conv.h
ACC_PROX_PERIOD, 79
ACC_PROX_SAMP_FREQ, 79
ACC_SAMP_NB, 79
ADCS_3_CHAN, 79
ADCS_5_CHAN, 79
ADCS_6_CHAN, 79
ALL_ADC, 79
e_ad_is_acquisition_completed, 79
e_ad_is_array_filled, 79
e_ad_scan_off, 80
e_ad_scan_on, 80
e_init_ad_scan, 80
MIC_SAMP_FREQ, 79
MIC_SAMP_NB, 79
MICRO_ONLY, 79
PULSE_LENGTH, 79
PULSE_PERIOD, 79

advance_ad_scan/e_micro.c
e_get_micro, 82
e_get_micro_average, 83
e_get_micro_last_values, 83
e_get_micro_volume, 83
e_last_micro_scan_id, 83
e_mic_scan, 83

advance_ad_scan/e_micro.h
e_get_micro, 86
e_get_micro_average, 87
e_get_micro_volume, 87
MIC0_BUFFER, 86
MIC1_BUFFER, 86
MIC2_BUFFER, 86

advance_ad_scan/e_prox.c
e_ambient_and_reflected_ir, 92
e_ambient_ir, 92
e_calibrate_ir, 91
e_get_ambient_light, 91
e_get_calibrated_prox, 91
e_get_prox, 91
init_value_ir, 92

advance_ad_scan/e_prox.h
e_calibrate_ir, 97
e_get_ambient_light, 97
e_get_calibrated_prox, 97
e_get_prox, 97

advance_one_timer/e_led.c
e_blink_led, 252
e_blink_led0, 252
e_blink_led1, 252
e_blink_led2, 252
e_blink_led3, 253
e_blink_led4, 253
e_blink_led5, 253
e_blink_led6, 253
e_blink_led7, 253
e_led_clear, 253
e_set_blinking_cycle, 254
e_set_body_led, 254
e_set_front_led, 254
e_set_led, 254
start_led_blinking, 255
stop_led_blinking, 255
flow_led, 255
k2000_led, 255
LED_EFFECTS, 252
left_led, 255
right_led, 255
snake_led, 255

advance_one_timer/e_led.h
e_blink_led, 264
e_blink_led0, 265
e_blink_led1, 265
e_blink_led2, 265
e_blink_led3, 265
e_blink_led4, 265
e_blink_led5, 265

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
e_blink_led6, 266
e_blink_led7, 266
e_led_clear, 266
e_set_body_led, 266
e_set_front_led, 266
e_set_led, 267
e_start_led_blinking, 267
e_stop_led_blinking, 267
flow_led, 268
k2000_led, 268
left_led, 268
right_led, 268
snake_led, 268

advance_one_timer/e_motors.c
  e_get_steps_left, 277
e_get_steps_right, 277
e_init_motors, 278
e_set_speed, 278
e_set_speed_left, 278
e_set_speed_right, 278
e_set_steps_left, 279
e_set_steps_right, 279
  left_motor_phase, 279
  left_speed, 279
MAXV, 277
nbr_steps_left, 279
nbr_steps_right, 279
POWERSAVE, 277
right_motor_phase, 279
right_speed, 279
run_left_motor, 279
run_right_motor, 279
TRESHV, 277

advance_one_timer/e_motors.h
  e_get_steps_left, 288
e_get_steps_right, 288
e_init_motors, 288
e_set_speed, 289
e_set_speed_left, 289
e_set_speed_right, 290
e_set_steps_left, 291
e_set_steps_right, 291

advance_one_timer/fast_agenda/e_led.c
  e_blink_led, 257
e_blink_led0, 257
e_blink_led1, 257
e_blink_led2, 258
e_blink_led3, 258
e_blink_led4, 258
e_blink_led5, 258
e_blink_led6, 258
e_blink_led7, 258
e_led_clear, 259
e_set_blinking_cycle, 259
  e_set_body_led, 259
  e_set_front_led, 259
  e_set_led, 259
  e_start_led_blinking, 260
  e_stop_led_blinking, 260

advance_one_timer/fast_agenda/e_motors.c
  e_get_steps_left, 281
e_get_steps_right, 281
e_init_motors, 281
e_set_speed, 282
e_set_speed_left, 282
e_set_speed_right, 282
e_set_steps_left, 282
e_set_steps_right, 283
  left_motor_phase, 283
  left_speed, 283
MAXV, 281
nbr_steps_left, 283
nbr_steps_right, 283
POWERSAVE, 281
right_motor_phase, 283
right_speed, 283
run_left_motor, 283
run_right_motor, 283
TRESHV, 281

advance_one_timer/fast_agenda/e_motors.h
  e_get_steps_left, 293
e_get_steps_right, 293
e_init_motors, 293
e_set_speed, 293
e_set_speed_left, 294
e_set_speed_right, 294
e_set_steps_left, 295
e_set_steps_right, 295
AESPEED_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
AG_ALREADY_CREATED
bpp_current
  slow_3_timer/e_timers.c, 194
BRICTR_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
BtDevice, 59
  address, 59
  class, 59
  friendly_name, 59
BtEPuck, 60
  address, 60
  number, 60
buf_pos
  slow_3_timer/e_timers.c, 194
buffer
  slow_3_timer/e_timers.c, 194
BUFFER_DATA_LENGTH
  ComModule.c, 211
bytes_per_pixel
  slow_3_timer/e_timers.c, 194
calculate_acc_raw
  e_acc.c, 65
calculate_acc_spherical
  e_acc.c, 65
CAM_DATA
  e_epuck_ports.h, 326
CAM_HREF
  e_epuck_ports.h, 326
CAM_HREF_DIR
  e_epuck_ports.h, 326
CAM_PCLK
  e_epuck_ports.h, 326
CAM_PCLK_DIR
  e_epuck_ports.h, 326
CAM_PWDN
  e_epuck_ports.h, 326
CAM_PWDN_DIR
  e_epuck_ports.h, 326
cam_reg
  fast_2_timer/e_registers.c, 172
  slow_3_timer/e_registers.c, 188
CAM_RESET
  e_epuck_ports.h, 326
CAM_RESET_DIR
  e_epuck_ports.h, 326
CAM_VSYNC
  e_epuck_ports.h, 326
CAM_VSYNC_DIR
  e_epuck_ports.h, 326
CAM_y0
  e_epuck_ports.h, 326
CAM_y0_DIR
  e_epuck_ports.h, 326
CAM_y1
  e_epuck_ports.h, 326
CAM_y1_DIR
  e_epuck_ports.h, 326
CAM_y2
  e_epuck_ports.h, 326
CAM_y2_DIR
  e_epuck_ports.h, 326
CAM_y3
  e_epuck_ports.h, 326
CAM_y3_DIR
  e_epuck_ports.h, 326
CAM_y4
  e_epuck_ports.h, 326
CAM_y4_DIR
  e_epuck_ports.h, 326
CAM_y5
  e_epuck_ports.h, 326
CAM_y5_DIR
  e_epuck_ports.h, 326
CAM_y6
  e_epuck_ports.h, 326
CAM_y6_DIR
  e_epuck_ports.h, 326
CAM_y7
  e_epuck_ports.h, 326
CAM_y7_DIR
  e_epuck_ports.h, 326
Camera fast two timers, 15
Camera slow three timers, 18
camera/ Directory Reference, 41
camera/fast_2_timer/ Directory Reference, 44
camera/fast_2_timer/e_calc.c, 121
camera/fast_2_timer/e_po3030k.h, 125
camera/fast_2_timer/e_registers.c, 157
camera/fast_2_timer/e_timers.c, 189
camera/slow_3_timer/ Directory Reference, 52
camera/slow_3_timer/e_calc.c, 123
camera/slow_3_timer/e_po3030k.h, 141
camera/slow_3_timer/e_registers.c, 173
camera/slow_3_timer/e_timers.c, 192
CBCRGAIN_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
centre_x
  e_acc.c, 68
centre_y
  e_acc.c, 68
centre_z
  e_acc.c, 68
CHAN_DOWN
  e_remote_control.h, 311
CHAN_UP
  e_remote_control.h, 311
check
e_remote_control.c, 308
check_temp
e_remote_control.c, 308
class
BtDevice, 59
CloseEpuck.m
EpuckPort, 231
close, 231
CLRWD T
e_epuck_ports.h, 326
codec/ Directory Reference, 42
codec/e_common.inc, 196
codec/e_sound.c, 197
codec/e_sound.h, 199
COLOR_GAIN_BASE
fast_2_timer/e_registers.c, 160
slow_3_timer/e_registers.c, 176
COLOR_COEF_BASE
fast_2_timer/e_registers.c, 160
slow_3_timer/e_registers.c, 176
COLOR_M_ADDR
fast_2_timer/e_registers.c, 160
slow_3_timer/e_registers.c, 176
COM_MODULE_DEFAULT_GROUP
ComModule.h, 213
COM_MODULE_HW_ATTENUATOR_0DB
ComModule.h, 213
COM_MODULE_HW_ATTENUATOR_25DB
ComModule.h, 213
COM_MODULE_I2C_ADDR
ComModule.c, 211
COM_MODULE_MAXSIZE
ComModule.h, 213
ComModule.c
ADDR_DATA_IN_PACKET_OFFSET, 211
ADDR_DATA_IN_PACKET_OFFSET, 211
AM_MSGTYPE, 211
AM_MSGTYPE_IN_PACKET_OFFSET, 211
BUFFER_DATA_LENGTH, 211
COM_MODULE_I2C_ADDR, 211
CONFIG_REG_ADDR, 211
FIRST_DATA_IN_PACKET_OFFSET, 211
GetHardwareAttenuator, 211
GetOwnAddress, 211
GetOwnGroup, 211
GetRadioEnabledState, 211
GetSoftwareAttenuator, 211
GetStatus, 211
InitComModule, 211
IsModulePlugged, 211
IsPacketReady, 211
compute_gcd
e_agenda_fast.c, 315
CONFIG_REG_ADDR
ComModule.c, 211
continue
EpuckGetData.m, 233
c contrib/ Directory Reference, 43
c contrib/LIS_sensors_turret/ Directory Reference, 48
OWNADDRH_REG_ADDR, 211
OWNADDRL_REG_ADDR, 211
OWNGROUP_REG_ADDR, 211
PACKET_LOST_FLAG, 211
PACKET_READY_FLAG, 211
 RADIO_ENABLED_FLAG, 211
ReadRegister, 211
REC_BUFFER_END, 211
REC_BUFFER_START, 211
REQUEST_TO_SEND_FLAG, 211
SEND_BUFFER_END, 211
SEND_BUFFER_START, 211
SEND_REG_ADDR, 211
SendPacket, 211
SetHardwareAttenuator, 211
SetOwnAddress, 211
SetOwnGroup, 211
SetRadioEnabledState, 211
SetSoftwareAttenuator, 211
SOFTATT_REG_ADDR, 211
STATUS_REG_ADDR, 211
TX_IDLE_FLAG, 211
TX_SEND_ERROR, 211
TYPEDATA_IN_PACKET_OFFSET, 211
WriteRegister, 211
contrib/ LIS_sensors_turret/ Directory Reference, 48
contrib/LIS_sensors_turret/ Directory Reference, 48
Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
| contrib/LIS_sensors_turret/e_devantech.c, 201 |
| contrib/LIS_sensors_turret/e_devantech.h, 202 |
| contrib/LIS_sensors_turret/e_sensext.c, 204 |
| contrib/LIS_sensors_turret/e_sensext.h, 205 |
| contrib/LIS_sensors_turret/e_sharp.c, 206 |
| contrib/LIS_sensors_turret/e_sharp.h, 207 |
| contrib/SWIS_com_module/ Directory Reference, 53 |
| contrib/SWIS_com_module/ComModule.c, 209 |
| contrib/SWIS_com_module/ComModule.h, 212 |
| AgendaType, 57 |
| CST_RADIAN |
| e_acc.h, 71 |
| current_col |
| slow_3_timer/e_timers.c, 194 |
| current_row |
| slow_3_timer/e_timers.c, 194 |
| cycle |
| AgendaType, 57 |
| data |
| e_remote_control.c, 308 |
| EpuckGetData.m, 233 |
| data_temp |
| e_remote_control.c, 308 |
| DEVICE_ID |
| fast_2_timer/e_registers.c, 160 |
| slow_3_timer/e_registers.c, 176 |
| e_acc.c |
| acc_x, 68 |
| acc_y, 68 |
| acc_z, 68 |
| acceleration, 68 |
| angle_mem, 68 |
| calculate_acc_raw, 65 |
| calculate_acc_spherical, 65 |
| centre_x, 68 |
| centre_y, 68 |
| centre_z, 68 |
| e_acc_calibr, 65 |
| e_acc_scan, 68 |
| e_get_acc, 65 |
| e_get_acc_filtered, 66 |
| e_last_acc_scan_id, 68 |
| e_read_acc, 66 |
| e_read_acc_spheric, 66 |
| e_read_acc_x, 66 |
| e_read_acc_xyz, 66 |
| e_read_acc_y, 67 |
| e_read_acc_z, 67 |
| e_read_inclination, 67 |
| e_read_orientation, 67 |
| inclination, 68 |
| orientation, 68 |
| e_acc.h |
| ACCX_BUFFER, 71 |
| ACCY_BUFFER, 71 |
| ACCZ_BUFFER, 71 |
| ANGLE_ERROR, 71 |
| CST_RADIAN, 71 |
| e_acc_calibr, 71 |
| e_display_angle, 71 |
| e_get_acc, 71 |
| e_get_acc_filtered, 71 |
| e_read_acc, 71 |
| e_read_acc_spheric, 72 |
| e_read_acc_x, 72 |
| e_read_acc_xyz, 72 |
| e_read_acc_y, 72 |
| e_read_acc_z, 72 |
| e_read_inclination, 73 |
| e_read_orientation, 73 |
| FILTER_SIZE, 71 |
| e_acc_calibr |
| e_acc.c, 65 |
| e_acc.h, 71 |
| e_acc_scan |
| advance_ad_scan/e_ad_conv.c, 76 |
| e_acc.c, 68 |
| e_accelerometer.c |
| e_get_acc, 103 |
| e_init_acc, 103 |
| e_accelerometer.h |
| e_get_acc, 104 |
| e_init_acc, 105 |
| e_activate_agenda |
| e_agenda.c, 242 |
| e_agenda.h, 246 |
| e_agenda_fast.c, 315 |
| e_agenda_fast.h, 319 |
| e_activate_motors |
| e_agenda_fast.c, 315 |
| e_agenda_fast.h, 319 |
| e_ad_conv.c |
| e_init_ad, 77 |
| e_read_ad, 77 |
| e_ad_conv.h |
| e_init_ad, 81 |
| e_read_ad, 81 |
| e_ad_is_acquisition_completed |
| advance_ad_scan/e_ad_conv.c, 75 |
| advance_ad_scan/e_ad_conv.h, 79 |
| e_ad_is_array_filled |
| advance_ad_scan/e_ad_conv.c, 75 |
| advance_ad_scan/e_ad_conv.h, 79 |
e_ad_scan_off
    advance_ad_scan/e_ad_conv.c, 75
    advance_ad_scan/e_ad_conv.h, 80

e_ad_scan_on
    advance_ad_scan/e_ad_conv.c, 75
    advance_ad_scan/e_ad_conv.h, 80

e_agenda.c
    __attribute__, 242
    agenda_list, 244
e_activate_agenda, 242
e_destroy_agenda, 242
e_end_agendas_processing, 243
e_pause_agenda, 243
e_reset_agenda, 243
e_restart_agenda, 244
e_set_agenda_cycle, 244
e_start_agendas_processing, 244
EXIT_OK, 242

e_agenda.h
    AG_ALREADY_CREATED, 246
    AG_NOT_FOUND, 246
    Agenda, 246
e_activate_agenda, 246
e_destroy_agenda, 246
e_end_agendas_processing, 247
e_pause_agenda, 247
e_reset_agenda, 247
e_restart_agenda, 248
e_set_agenda_cycle, 248
e_start_agendas_processing, 249

e_agenda_fast.c
    __attribute__, 314
    agenda_list, 317
    assign_agenda, 314
    compute_gcd, 315
e_activate_agenda, 315
e_activate_motors, 315
e_configure_timer, 315
e_destroy_agenda, 315
e_end_agendas_processing, 315
e_pause_agenda, 316
e_reset_agenda, 316
e_restart_agenda, 316
e_set_agenda_cycle, 316
e_set_motor_speed, 316
e_set_timer_speed, 316
e_start_agendas_processing, 316
e_start_timer_processing, 316
EXIT_OK, 314
find_function, 317
migrate, 317
my ceil, 317
recompute_speeds, 317
search_best_fit, 317

e_agenda_fast.h
    AG_ALREADY_CREATED, 319
    AG_NOT_FOUND, 319
    Agenda, 319
e_activate_agenda, 319
e_activate_motors, 319
e_configure_timer, 320
e_destroy_agenda, 320
e_end_agendas_processing, 320
e_pause_agenda, 320
e_reset_agenda, 321
e_restart_agenda, 321
e_set_agenda_cycle, 322
e_set_motor_speed, 322
e_start_agendas_processing, 322
e_start_timer_processing, 322
e_ambient_and_reflected_ir
    advance_ad_scan/e_ad_conv.c, 76
    advance_ad_scan/e_prox.c, 92

e_ambient_ir
    advance_ad_scan/e_ad_conv.c, 76
    advance_ad_scan/e_prox.c, 92

e_blink_led
    advance_one_timer/e_led.c, 252
    advance_one_timer/e_led.h, 264
    advance_one_timer/fast_agenda/e_led.c, 257
    advance_one_timer/fast_agenda/e_led.h, 270

e_blink_led0
    advance_one_timer/e_led.c, 252
    advance_one_timer/e_led.h, 265
    advance_one_timer/fast_agenda/e_led.c, 257
    advance_one_timer/fast_agenda/e_led.h, 270

e_blink_led1
    advance_one_timer/e_led.c, 252
    advance_one_timer/e_led.h, 265
    advance_one_timer/fast_agenda/e_led.c, 257
    advance_one_timer/fast_agenda/e_led.h, 270

e_blink_led2
    advance_one_timer/e_led.c, 252
    advance_one_timer/e_led.h, 265
    advance_one_timer/fast_agenda/e_led.c, 258
    advance_one_timer/fast_agenda/e_led.h, 270

e_blink_led3
    advance_one_timer/e_led.c, 253
    advance_one_timer/e_led.h, 265
    advance_one_timer/fast_agenda/e_led.c, 258
    advance_one_timer/fast_agenda/e_led.h, 271

e_blink_led4
    advance_one_timer/e_led.c, 253
    advance_one_timer/e_led.h, 265
    advance_one_timer/fast_agenda/e_led.c, 258
    advance_one_timer/fast_agenda/e_led.h, 271

e_blink_led5
    advance_one_timer/e_led.c, 253
advance_one_timer/e_led.h, 265
advance_one_timer/fast_agenda/e_led.c, 258
advance_one_timer/fast_agenda/e_led.h, 271
e_blink_led6
  advance_one_timer/e_led.c, 253
  advance_one_timer/e_led.h, 266
  advance_one_timer/fast_agenda/e_led.c, 258
  advance_one_timer/fast_agenda/e_led.h, 271
e_blink_led7
  advance_one_timer/e_led.c, 253
  advance_one_timer/e_led.h, 266
  advance_one_timer/fast_agenda/e_led.c, 258
  advance_one_timer/fast_agenda/e_led.h, 271
e_bluetooth.c
  e_bt_connect_epuck, 110
  e_bt_establish_SPP_link, 110
  e_bt_exit_transparent_mode, 110
  e_bt_factory_reset, 110
  e_bt_get_friendly_name, 111
  e_bt_inquiry, 111
  e_bt_list_local_paired_device, 111
  e_bt_local_paired_device, 114
  e_bt_present_device, 114
  e_bt_present_epuck, 114
  e_bt_read_local_name, 112
  e_bt_read_local_pin_number, 112
  e_bt_release_SPP_link, 112
  e_bt_remove_local_paired_device, 112
  e_bt_reset, 112
  e_bt_send_SPP_data, 113
  e_bt_transparent_mode, 113
  e_bt_write_local_name, 113
  e_bt_write_local_pin_number, 113
local_bt_PIN, 114
e_bluetooth.h
  e_bt_connect_epuck, 116
  e_bt_establish_SPP_link, 116
  e_bt_exit_transparent_mode, 116
  e_bt_factory_reset, 116
  e_bt_get_friendly_name, 117
  e_bt_inquiry, 117
  e_bt_list_local_paired_device, 117
  e_bt_local_paired_device, 120
  e_bt_present_device, 120
  e_bt_present_epuck, 120
  e_bt_read_local_name, 120
  e_bt_read_local_pin_number, 120
  e_bt_release_SPP_link, 120
  e_bt_remove_local_paired_device, 120
  e_bt_reset, 120
  e_bt_send_SPP_data, 120
  e_bt_transparent_mode, 120
e_bluetooth.c, 113
e_bluetooth.h, 119
e_bt_write_local_name
e_bluetooth.c, 113
  e_bluetooth.h, 119
  e_bt_write_local_pin_number
e_bluetooth.c, 113
  e_bluetooth.h, 119
  e_calibrate_ir
    advance_ad_scan/e_prox.c, 91
    advance_ad_scan/e_prox.h, 97
e_close_sound
e_sound.c, 197
  e_sound.h, 199
e_configure_timer
e_agenda_fast.c, 315
  e_agenda_fast.h, 320
e_destroy_agenda
e_agenda.c, 242
  e_agenda.h, 246
  e_agenda_fast.c, 315
  e_agenda_fast.h, 320
e_devantech.c
e_disable_devantech, 201
  e_get_delay_devantech, 201
  e_get_dist_devantech, 201
  e_get_light_devantech, 201
  e_get_sr_devantech, 201
  e_i2cd_readb, 201
  e_i2cd_write, 201
  e_init_devantech, 201
  e_set_gain_devantech, 201
  e_set_range_devantech, 201
  e_devantech.h
    e_disable_devantech, 203
    e_get_delay_devantech, 203
    e_get_dist_devantech, 203
    e_get_light_devantech, 203
    e_get_sr_devantech, 203
    e_i2cd_readb, 203
    e_i2cd_readw, 203
    e_i2cd_write, 203
    e_init_devantech, 203
    e_set_gain_devantech, 203
    e_set_range_devantech, 203
e_disable_devantech
e_devantech.c, 201
  e_devantech.h, 203
e_display_angle
e_acc.c, 65
  e_acc.h, 71
e_doFFT_asm
e_fft.c, 214
  e_fft.h, 215
  e_end_agendas_processing
    e_agenda.c, 243
    e_agenda.h, 247
    e_agenda_fast.c, 315
    e_agenda_fast.h, 320
  e_epuck_ports.h
    ACCX, 326
    ACCY, 326
    ACCZ, 326
    AUDIO_ON, 326
    AUDIO_ON_DIR, 326
    BATT_LOW, 326
    BATT_LOW_DIR, 326
    BODY_LED, 326
    BODY_LED_DIR, 326
    CAM_DATA, 326
    CAM_HREF, 326
    CAM_HREF_DIR, 326
    CAM_PCLK, 326
    CAM_PCLK_DIR, 326
    CAM_PWDN, 326
    CAM_PWDN_DIR, 326
    CAM_RESET, 326
    CAM_RESET_DIR, 326
    CAM_VSYNC, 326
    CAM_VSYNC_DIR, 326
    CAM_y0, 326
    CAM_y0_DIR, 326
    CAM_y1, 326
    CAM_y1_DIR, 326
    CAM_y2, 326
    CAM_y2_DIR, 326
    CAM_y3, 326
    CAM_y3_DIR, 326
    CAM_y4, 326
    CAM_y4_DIR, 326
    CAM_y5, 326
    CAM_y5_DIR, 326
    CAM_y6, 326
    CAM_y6_DIR, 326
    CAM_y7, 326
    CAM_y7_DIR, 326
    CLRWDT, 326
    FALSE, 326
    FCY, 326
    FOSC, 326
    FRONT_LED, 326
    FRONT_LED_DIR, 326
    IDLE, 326
    INPUT_PIN, 326
    INTERRUPT_DELAY, 326
    INTERRUPT_OFF, 326
    INTERRUPT_ON, 326
    IR0, 326
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IR1</td>
<td>326</td>
<td>PULSE_IR3, 326</td>
</tr>
<tr>
<td>IR2</td>
<td>326</td>
<td>PULSE_IR3_DIR, 326</td>
</tr>
<tr>
<td>IR3</td>
<td>326</td>
<td>REMOTE, 326</td>
</tr>
<tr>
<td>IR4</td>
<td>326</td>
<td>REMOTE_DIR, 326</td>
</tr>
<tr>
<td>IR5</td>
<td>326</td>
<td>RESET, 326</td>
</tr>
<tr>
<td>IR6</td>
<td>326</td>
<td>SELECTOR0, 326</td>
</tr>
<tr>
<td>IR7</td>
<td>326</td>
<td>SELECTOR0_DIR, 326</td>
</tr>
<tr>
<td>LED0</td>
<td>326</td>
<td>SELECTOR1, 326</td>
</tr>
<tr>
<td>LED0_DIR</td>
<td>326</td>
<td>SELECTOR1_DIR, 326</td>
</tr>
<tr>
<td>LED1</td>
<td>326</td>
<td>SELECTOR2, 326</td>
</tr>
<tr>
<td>LED1_DIR</td>
<td>326</td>
<td>SELECTOR2_DIR, 326</td>
</tr>
<tr>
<td>LED2</td>
<td>326</td>
<td>SELECTOR3, 326</td>
</tr>
<tr>
<td>LED2_DIR</td>
<td>326</td>
<td>SELECTOR3_DIR, 326</td>
</tr>
<tr>
<td>LED3</td>
<td>326</td>
<td>SIO_C, 326</td>
</tr>
<tr>
<td>LED3_DIR</td>
<td>326</td>
<td>SIO_C_DIR, 326</td>
</tr>
<tr>
<td>LED4</td>
<td>326</td>
<td>SIO_D, 326</td>
</tr>
<tr>
<td>LED4_DIR</td>
<td>326</td>
<td>SIO_D_DIR, 326</td>
</tr>
<tr>
<td>LED5</td>
<td>326</td>
<td>SLEEP, 326</td>
</tr>
<tr>
<td>LED5_DIR</td>
<td>326</td>
<td>STOP_TMR1, 326</td>
</tr>
<tr>
<td>LED6</td>
<td>326</td>
<td>STOP_TMR2, 326</td>
</tr>
<tr>
<td>LED6_DIR</td>
<td>326</td>
<td>STOP_TMR3, 326</td>
</tr>
<tr>
<td>LED7</td>
<td>326</td>
<td>STOP_TMR4, 326</td>
</tr>
<tr>
<td>LED7_DIR</td>
<td>326</td>
<td>STOP_TMR5, 326</td>
</tr>
<tr>
<td>MIC1</td>
<td>326</td>
<td>TCY PIC, 326</td>
</tr>
<tr>
<td>MIC2</td>
<td>326</td>
<td>TRUE, 326</td>
</tr>
<tr>
<td>MIC3</td>
<td>326</td>
<td>e_fast_copy</td>
</tr>
<tr>
<td>MICROSEC</td>
<td>326</td>
<td>e_fft_utilities.h, 216</td>
</tr>
<tr>
<td>MILLISEC</td>
<td>326</td>
<td>e_fft.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>attribute</strong>, 214</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_doFFT_asm, 214</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_fft.h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_fft_asm, 215</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FFT_BLOCK_LENGTH, 215</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOG2_BLOCK_LENGTH, 215</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_fft_utilities.h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_fast_copy, 216</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_subtract_mean, 216</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_get_acc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_acc.c, 65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_acc.h, 71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_accelerometer.c, 103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_accelerometer.h, 104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_get_acc_filtered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_acc.c, 66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_acc.h, 71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_get_address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_remote_control.c, 307</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_remote_control.h, 311</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_get_ambient_light</td>
</tr>
<tr>
<td></td>
<td></td>
<td>advance_ad_scan/e_prox.c, 91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>advance_ad_scan/e_prox.h, 97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_prox.c, 94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_prox.h, 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e_prox_timer2.c, 107</td>
</tr>
</tbody>
</table>
INDEX

350

e_i2c_reset, 225
e_i2c_restart, 225
e_i2c_start, 225
e_i2c_stop, 225
e_i2c_write, 226
ERROR, 224
OPERATION_OK, 224
READ, 224
RESTART, 224
START, 224
STOP, 224
WRITE, 224
e_i2c_mode
  e_I2C_master_module.c, 222
e_i2c_nack
  e_I2C_master_module.c, 221
  e_I2C_master_module.h, 225
e_I2C_protocol.c
  e_i2cp_disable, 227
  e_i2cp_enable, 227
  e_i2cp_init, 228
  e_i2cp_read, 228
  e_i2cp_write, 228
e_I2C_protocol.h
  e_i2cp_disable, 229
  e_i2cp_enable, 230
  e_i2cp_init, 230
  e_i2cp_read, 230
  e_i2cp_read_string, 230
  e_i2cp_write, 230
  e_i2cp_write_string, 230
e_i2c_read
  e_I2C_master_module.c, 221
  e_I2C_master_module.h, 225
e_i2c_reset
  e_I2C_master_module.c, 221
  e_I2C_master_module.h, 225
e_i2c_restart
  e_I2C_master_module.c, 221
  e_I2C_master_module.h, 225
e_i2c_start
  e_I2C_master_module.c, 221
  e_I2C_master_module.h, 225
e_i2c_stop
  e_I2C_master_module.c, 221
  e_I2C_master_module.h, 225
e_i2c_write
  e_I2C_master_module.c, 222
  e_I2C_master_module.h, 226
e_i2cd_readb
  e_devantech.c, 201
e_devantech.h, 203
e_i2cd_write
e_devantech.c, 201
e_devantech.h, 203
e_i2cp_disable
  e_I2C_protocol.c, 227
  e_I2C_protocol.h, 229
e_i2cp_enable
  e_I2C_protocol.c, 227
  e_I2C_protocol.h, 230
e_i2cp_init
  e_I2C_protocol.c, 228
  e_I2C_protocol.h, 230
e_i2cp_read
  e_I2C_protocol.c, 228
  e_I2C_protocol.h, 230
e_i2cp_read_string
  e_I2C_protocol.h, 230
e_i2cp_write
  e_I2C_protocol.c, 228
  e_I2C_protocol.h, 230
e_i2cp_write_string
  e_I2C_protocol.h, 230
e_init_acc
  e_accelerometer.c, 103
  e_accelerometer.h, 105
e_init_ad
  e_ad_conv.c, 77
  e_ad_conv.h, 81
e_init_ad_scan
  advance_ad_scan/e_ad_conv.c, 75
  advance_ad_scan/e_ad_conv.h, 80
e_init_codec_slave
  e_sound.h, 199
e_init_dci_master
  e_sound.h, 199
e_init_devantech
  e_devantech.c, 201
  e_devantech.h, 203
e_init_micro
  e_micro.c, 85
  e_micro.h, 89
e_init_motors
  advance_one_timer/e_motors.c, 278
  advance_one_timer/e_motors.h, 288
  advance_one_timer/fast_agenda/e_motors.c, 281
  advance_one_timer/fast_agenda/e_motors.h, 293
e_motors.c, 285
e_motors.h, 298
e_motors_kml.c, 303
e_motors_timer3.c, 330
e_init_port
e_init_port.c, 327

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
INDEX

e_init_port.h, 328
e_init_port.c
  _FBORPOR, 327
  FGS, 327
  FOSC, 327
  _FWDT, 327
  e_init_port, 327

e_init_port.h
  e_init_port, 328

e_init_prox
  e_prox.c, 95
  e_prox.h, 101
  e_prox_timer2.c, 107

e_init_remote_control
  e_remote_control.c, 307
  e_remote_control.h, 311

e_init_sensext
  e_sensext.c, 204
  e_sensext.h, 205

e_init_sharp
  e_sharp.c, 206
  e_sharp.h, 208

e_init_sound
  e_sound.c, 197
  e_sound.h, 199

e_init_uart1
  e_uart_char.h, 336

e_init_uart2
  e_uart_char.h, 336

e_input_signal.c
  __attribute__, 217

e_interrupts
  e_I2C_master_module.c, 222

e_ischar_uart1
  e_uart_char.h, 336

e_ischar_uart2
  e_uart_char.h, 336

e_last_acc_scan_id
  advance_ad_scan/e_ad_conv.c, 76
  e_acc.c, 68

e_last_mic_scan_id
  advance_ad_scan/e_ad_conv.c, 76
  advance_ad_scan/e_micro.c, 83

e_led.c
  e_led_clear, 262
  e_set_body_led, 262
  e_set_front_led, 262
  e_set_led, 262

e_led.h
  e_led_clear, 274
  e_set_body_led, 275
  e_set_front_led, 275
  e_set_led, 275

e_led_clear

advance_one_timer/e_led.c, 253
advance_one_timer/e_led.h, 266
advance_one_timer/fast_agenda/e_led.c, 259
advance_one_timer/fast_agenda/e_led.h, 272
  e_led.c, 262
  e_led.h, 274

e_micro.c
  e_get_micro, 84
  e_init_micro, 85

e_micro.h
  e_get_micro, 88
  e_init_micro, 89

e_motors.c
  __attribute__, 285
  e_get_steps_left, 285
  e_get_steps_right, 285
  e_init_motors, 285
  e_set_speed_left, 285
  e_set_speed_right, 286
  e_set_steps_left, 286
  e_set_steps_right, 286
  left_speed, 286
  nbr_pas_left, 286
  nbr_pas_right, 286
  right_speed, 286

e_motors.h
  e_get_steps_left, 298
  e_get_steps_right, 298
  e_init_motors, 298
  e_set_speed_left, 298
  e_set_speed_right, 299
  e_set_steps_left, 300
  e_set_steps_right, 300

e_motors_kml.c
  e_get_steps_left, 302
  e_get_steps_right, 302
  e_init_motors, 303
  e_set_speed, 303
  e_set_speed_left, 303
  e_set_speed_right, 303
  e_set_steps_left, 304
  e_set_steps_right, 304
  goal_active_left, 305
  goal_active_right, 305
  goal_steps_left, 305
  goal_steps_right, 305
  left_motor_phase, 305
  left_speed, 305
  MAXV, 302
  nbr_steps_left, 305
  nbr_steps_right, 305

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
POWERSAVE, 302
right_motor_phase, 305
right_speed, 305
run_left_motor, 304
run_right_motor, 304
TRESHV, 302
e_motors_timer3.c
__attribute__, 330
e_get_steps_left, 330
e_get_steps_right, 330
e_init_motors, 330
e_set_speed_left, 331
e_set_speed_right, 331
e_set_steps_left, 331
left_speed, 332
motor_counter_left, 332
motor_counter_left_init, 332
motor_counter_right, 332
motor_counter_right_init, 332
nbr_pas_left, 332
nbr_pas_right, 332
right_speed, 332
e_pause_agenda
e_agenda.c, 243
e_agenda.h, 247
e_agenda_fast.c, 316
e_agenda_fast.h, 320
e_play_sound
e_sound.c, 197
e_sound.h, 199
e_po3030k_apply_timer_config
fast_2_timer/e_po3030k.h, 127
fast_2_timer/e_timers.c, 189
slow_3_timer/e_po3030k.h, 143
slow_3_timer/e_timers.c, 193
e_po3030k_config_cam
fast_2_timer/e_calc.c, 121
fast_2_timer/e_po3030k.h, 128
slow_3_timer/e_calc.c, 123
slow_3_timer/e_po3030k.h, 144
e_po3030k_get_bytes_per_pixel
fast_2_timer/e_calc.c, 122
fast_2_timer/e_po3030k.h, 129
slow_3_timer/e_calc.c, 124
slow_3_timer/e_po3030k.h, 145
e_po3030k_get_register
fast_2_timer/e_po3030k.h, 129
fast_2_timer/e_registers.c, 160
slow_3_timer/e_po3030k.h, 145
slow_3_timer/e_registers.c, 176
e_po3030k_init_cam
fast_2_timer/e_calc.c, 122
fast_2_timer/e_po3030k.h, 129
slow_3_timer/e_calc.c, 124
slow_3_timer/e_po3030k.h, 145
e_po3030k_is_img_ready
fast_2_timer/e_po3030k.h, 129
fast_2_timer/e_timers.c, 190
slow_3_timer/e_po3030k.h, 145
slow_3_timer/e_timers.c, 193
e_po3030k_launch_capture
fast_2_timer/e_po3030k.h, 130
fast_2_timer/e_timers.c, 190
slow_3_timer/e_po3030k.h, 146
slow_3_timer/e_timers.c, 193
e_po3030k_read_cam_registers
fast_2_timer/e_po3030k.h, 130
fast_2_timer/e_registers.c, 161
slow_3_timer/e_po3030k.h, 146
slow_3_timer/e_registers.c, 177
e_po3030k_set_adc_offset
fast_2_timer/e_po3030k.h, 130
fast_2_timer/e_registers.c, 161
slow_3_timer/e_po3030k.h, 146
slow_3_timer/e_registers.c, 177
e_po3030k_set_ae_speed
fast_2_timer/e_po3030k.h, 130
fast_2_timer/e_registers.c, 161
slow_3_timer/e_po3030k.h, 146
slow_3_timer/e_registers.c, 177
e_po3030k_set_awb_ae
fast_2_timer/e_po3030k.h, 130
fast_2_timer/e_registers.c, 161
slow_3_timer/e_po3030k.h, 146
slow_3_timer/e_registers.c, 177
e_po3030k_set_awb_ae_tol
fast_2_timer/e_po3030k.h, 131
fast_2_timer/e_registers.c, 162
slow_3_timer/e_po3030k.h, 147
slow_3_timer/e_registers.c, 178
e_po3030k_set_bias
fast_2_timer/e_po3030k.h, 131
fast_2_timer/e_registers.c, 162
slow_3_timer/e_po3030k.h, 147
slow_3_timer/e_registers.c, 178
e_po3030k_set_brigh_contr
fast_2_timer/e_po3030k.h, 131
fast_2_timer/e_registers.c, 162
slow_3_timer/e_po3030k.h, 147
slow_3_timer/e_registers.c, 178
e_po3030k_set_cb_cr_gain
fast_2_timer/e_po3030k.h, 131
fast_2_timer/e_registers.c, 162
slow_3_timer/e_po3030k.h, 147
slow_3_timer/e_registers.c, 178
e_po3030k_set_color_gain
fast_2_timer/e_po3030k.h, 132
fast_2_timer/e_registers.c, 163
slow_3_timer/e_po3030k.h, 148
slow_3_timer/e_registers.c, 179
e_po3030k_set_color_matrix
  fast_2_timer/e_po3030k.h, 132
  slow_3_timer/e_po3030k.h, 148
e_po3030k_set_color_mode
  fast_2_timer/e_po3030k.h, 132
  fast_2_timer/e_registers.c, 163
  slow_3_timer/e_po3030k.h, 148
  slow_3_timer/e_registers.c, 179
e_po3030k_set_edge_prop
  fast_2_timer/e_po3030k.h, 132
  fast_2_timer/e_registers.c, 163
  slow_3_timer/e_po3030k.h, 148
  slow_3_timer/e_registers.c, 179
e_po3030k_set_exposure
  fast_2_timer/e_po3030k.h, 133
  fast_2_timer/e_registers.c, 164
  slow_3_timer/e_po3030k.h, 149
  slow_3_timer/e_registers.c, 180
e_po3030k_set_flicker_detection
  fast_2_timer/e_po3030k.h, 133
  fast_2_timer/e_registers.c, 164
  slow_3_timer/e_po3030k.h, 149
  slow_3_timer/e_registers.c, 180
e_po3030k_set_flicker_man_set
  fast_2_timer/e_po3030k.h, 133
  fast_2_timer/e_registers.c, 164
  slow_3_timer/e_po3030k.h, 149
  slow_3_timer/e_registers.c, 180
e_po3030k_set_flicker_mode
  fast_2_timer/e_po3030k.h, 134
  fast_2_timer/e_registers.c, 165
  slow_3_timer/e_po3030k.h, 150
  slow_3_timer/e_registers.c, 181
e_po3030k_set_gamma_coef
  fast_2_timer/e_po3030k.h, 134
  fast_2_timer/e_registers.c, 165
  slow_3_timer/e_po3030k.h, 150
  slow_3_timer/e_registers.c, 181
e_po3030k_set_integ_time
  fast_2_timer/e_po3030k.h, 134
  fast_2_timer/e_registers.c, 165
  slow_3_timer/e_po3030k.h, 150
  slow_3_timer/e_registers.c, 181
e_po3030k_set_lens_gain
  fast_2_timer/e_po3030k.h, 135
  fast_2_timer/e_registers.c, 166
  slow_3_timer/e_po3030k.h, 151
  slow_3_timer/e_registers.c, 182
e_po3030k_set_max_min_awb
  fast_2_timer/e_po3030k.h, 135
  fast_2_timer/e_registers.c, 166
e_po3030k_set_max_min_exp
  fast_2_timer/e_po3030k.h, 135
  fast_2_timer/e_registers.c, 166
  slow_3_timer/e_po3030k.h, 151
  slow_3_timer/e_registers.c, 182
e_po3030k_set_mirror
  fast_2_timer/e_po3030k.h, 136
  fast_2_timer/e_registers.c, 167
  slow_3_timer/e_po3030k.h, 152
  slow_3_timer/e_registers.c, 183
e_po3030k_set_ref_exposure
  fast_2_timer/e_po3030k.h, 136
  fast_2_timer/e_registers.c, 167
  slow_3_timer/e_po3030k.h, 152
  slow_3_timer/e_registers.c, 183
e_po3030k_set_register
  fast_2_timer/e_po3030k.h, 136
  fast_2_timer/e_registers.c, 167
  slow_3_timer/e_po3030k.h, 152
  slow_3_timer/e_registers.c, 183
e_po3030k_set_sampling_mode
  fast_2_timer/e_po3030k.h, 137
  fast_2_timer/e_registers.c, 168
  slow_3_timer/e_po3030k.h, 153
  slow_3_timer/e_registers.c, 184
e_po3030k_set_sepia
  fast_2_timer/e_po3030k.h, 137
  fast_2_timer/e_registers.c, 168
  slow_3_timer/e_po3030k.h, 153
  slow_3_timer/e_registers.c, 184
e_po3030k_set_sepia_tone
  fast_2_timer/e_po3030k.h, 137
  fast_2_timer/e_registers.c, 168
  slow_3_timer/e_po3030k.h, 153
  slow_3_timer/e_registers.c, 184
e_po3030k_set_speed
  fast_2_timer/e_po3030k.h, 137
  fast_2_timer/e_registers.c, 168
  slow_3_timer/e_po3030k.h, 153
  slow_3_timer/e_registers.c, 184
e_po3030k_set_vsync
  fast_2_timer/e_po3030k.h, 138
  fast_2_timer/e_registers.c, 169
  slow_3_timer/e_po3030k.h, 154
  slow_3_timer/e_registers.c, 185
e_po3030k_set_weight_win
  fast_2_timer/e_po3030k.h, 138
  fast_2_timer/e_registers.c, 169
  slow_3_timer/e_po3030k.h, 154
  slow_3_timer/e_registers.c, 185
e_po3030k_set_ww
  fast_2_timer/e_po3030k.h, 139
CHAN_DOWN, 311
CHAN_UP, 311
e_get_address, 311
e_get_check, 311
e_get_data, 311
e_init_remote_control, 311
e_read_remote_control, 311
I_II, 311
MUTE, 311
OUT_AUX_1, 311
STANDBY, 311
VOL_DOWN, 311
VOL_UP, 311
e_reset_agenda
e_agenda.c, 243
e_agenda.h, 247
e_agenda_fast.c, 316
e_agenda_fast.h, 321
e_restart_agenda
e_agenda.c, 244
e_agenda.h, 248
e_agenda_fast.c, 316
e_agenda_fast.h, 321
e_send_char_to_matlab
matlab.c, 238
matlab.h, 240
e_send_int_to_matlab
matlab.c, 238
matlab.h, 240
e_send_uart1_char
e_uart_char.h, 336
e_uart_char.h, 336
e_send_uart2_char
e_uart_char.h, 336
e_sensext.c
e_get_sensext_wait, 204
e_init_sensext, 204
e_sensext_process, 204
e_start_sensext_wait, 204
e_stop_sensext_wait, 204
e_sensext_wait, 204
e_sensext.h
e_get_sensext_wait, 205
e_init_sensext, 205
e_sensext_process, 205
e_start_sensext_wait, 205
e_stop_sensext_wait, 205
I2C_ADDR_CMP03, 205
I2C_ADDR_SENSEXT, 205
I2C_ADDR_SRF08, 205
I2C_ADDR_SRF10, 205
I2C_ADDR_SRF235, 205
e_sensext_process
e_sensext.c, 204
e_sensext.h, 205
e_set_agenda_cycle
e_agenda.c, 244
e_agenda.h, 248
e_agenda_fast.c, 316
e_agenda_fast.h, 322
e_set_blinking_cycle
advance_one_timer/e_led.c, 254
advance_one_timer/fast_agenda/e_led.c, 259
e_set_body_led
advance_one_timer/e_led.c, 254
advance_one_timer/e_led.h, 266
advance_one_timer/fast_agenda/e_led.c, 259
advance_one_timer/fast_agenda/e_led.h, 272
e_led.c, 262
e_led.h, 275
e_set_front_led
advance_one_timer/e_led.c, 254
advance_one_timer/e_led.h, 266
advance_one_timer/fast_agenda/e_led.c, 259
advance_one_timer/fast_agenda/e_led.h, 272
e_led.c, 262
e_led.h, 275
e_set_gain_devantech
e_devantech.c, 201
e_devantech.h, 203
e_set_led
advance_one_timer/e_led.c, 254
advance_one_timer/e_led.h, 267
advance_one_timer/fast_agenda/e_led.c, 259
advance_one_timer/fast_agenda/e_led.h, 272
e_led.c, 262
e_led.h, 275
e_set_motor_speed
e_agenda_fast.c, 316
e_agenda_fast.h, 322
e_set_range_devantech
e_devantech.c, 201
e_devantech.h, 203
e_set_sharp_led
e_sharp.c, 206
e_sharp.h, 208
e_set_speed
advance_one_timer/e_motors.c, 278
advance_one_timer/e_motors.h, 289
advance_one_timer/fast_agenda/e_motors.c, 282
advance_one_timer/fast_agenda/e_motors.h, 293
e_motors_kml.c, 303
e_set_speed_left
advance_one_timer/e_motors.c, 278
advance_one_timer/e_motors.h, 289
advance_one_timer/fast_agenda/e_motors.c, 282
advance_one_timer/fast_agenda/e_motors.h, e_motors.c, 285 e_motors.h, 298 e_motors_kml.c, 303 e_motors_timer3.c, 330
e_set_speed_right
  advance_one_timer/e_motors.c, 278 advance_one_timer/e_motors.h, 290 advance_one_timer/fast_agenda/e_motors.c, 282 advance_one_timer/fast_agenda/e_motors.h, 294 e_motors.c, 286 e_motors.h, 299 e_motors_kml.c, 303 e_motors_timer3.c, 331
e_set_steps_left
  advance_one_timer/e_motors.c, 279 advance_one_timer/e_motors.h, 291 advance_one_timer/fast_agenda/e_motors.c, 282 advance_one_timer/fast_agenda/e_motors.h, 295 e_motors.c, 286 e_motors.h, 300 e_motors_kml.c, 304 e_motors_timer3.c, 331
e_set_steps_right
  advance_one_timer/e_motors.c, 279 advance_one_timer/e_motors.h, 291 advance_one_timer/fast_agenda/e_motors.c, 283 advance_one_timer/fast_agenda/e_motors.h, 295 e_motors.c, 286 e_motors.h, 300 e_motors_kml.c, 304 e_motors_timer3.c, 331
e_set_timer_speed
e_agenda_fast.c, 316
e_sharp.c
  e_get_dist_sharp, 206 e_init_sharp, 206 e_set_sharp_led, 206 e_sharp_led_clear, 206 e_sharp_off, 206 e_sharp_on, 206
e_sharp.h
  e_get_dist_sharp, 208 e_init_sharp, 208 e_set_sharp_led, 208 e_sharp_led_clear, 208 e_sharp_off, 208 e_sharp_on, 208 SHARP, 208 SHARP_LED1, 208 SHARP_LED1_DIR, 208 SHARP_LED2, 208 SHARP_LED2_DIR, 208 SHARP_LED3, 208 SHARP_LED3_DIR, 208 SHARP_LED4, 208 SHARP_LED4_DIR, 208 SHARP_LED5, 208 SHARP_LED5_DIR, 208 SHARP_VIN, 208 SHARP_VIN_DIR, 208
e_sharp_led_clear
e_sharp.c, 206 e_sharp.h, 208
e_sharp_off
e_sharp.c, 206 e_sharp.h, 208
e_sharp_on
e_sharp.c, 206 e_sharp.h, 208
e_sound.c
e_close_sound, 197 e_init_sound, 197 e_play_sound, 197
e_sound.h
  e_close_sound, 199 e_init_codec_slave, 199 e_init_dci_master, 199 e_init_sound, 199 e_play_sound, 199 e_sub_dci_kickoff, 200
e_start_agendas_processing
e_agenda.c, 244 e_agenda.h, 249 e_agenda_fast.c, 316 e_agenda_fast.h, 322
e_start_blinking
  advance_one_timer/e_led.c, 255 advance_one_timer/e_led.h, 267 advance_one_timer/fast_agenda/e_led.c, 260 advance_one_timer/fast_agenda/e_led.h, 273
e_start_sensext_wait
e_sensext.c, 204 e_sensext.h, 205
e_start_processing
e_agenda_fast.c, 316 e_agenda_fast.h, 322
e_stop_blinking
  advance_one_timer/e_led.c, 255 advance_one_timer/e_led.h, 267 advance_one_timer/fast_agenda/eLed.c, 260

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
INDEX

advance_one_timer/fast_agenda/eLed.h, 273
e_stop_prox
  e_prox.c, 95
e_prox.h, 101
e_prox_timer2.c, 107
e_stop_sensext_wait
  e_sensext.c, 204
e_sensext.h, 205
e_sub_dci_kickoff
  e_sound.h, 200
e_subtract_mean
  e_fft_utilities.h, 216
e_twiddle_factors.c
    __attribute__, 218
e_uart1_int_clr_addr
  e_uart_char.h, 337
e_uart1_int_clr_mask
  e_uart_char.h, 337
e_uart1_sending
  e_uart_char.h, 336
e_uart2_int_clr_addr
  e_uart_char.h, 337
e_uart2_int_clr_mask
  e_uart_char.h, 337
e_uart2_sending
  e_uart_char.h, 337
e_uart_char.h
  e_getchar_uart1, 335
e_getchar_uart2, 335
e_init_uart1, 336
e_init_uart2, 336
e_ischar_uart1, 336
e_ischar_uart2, 336
e_send_uart1_char, 336
e_send_uart2_char, 336
e_uart1_int_clr_addr, 337
e_uart1_int_clr_mask, 337
e_uart1_sending, 336
e_uart2_int_clr_addr, 337
e_uart2_int_clr_mask, 337
e_uart2_sending, 337

EDGE_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
elseif
  EpuckGetData.m, 233
EpuckFlush.m
  EpuckPort, 232
flush, 232
flushinput, 232
flushoutput, 232
function, 232
EpuckGetData.m
  continue, 233
data, 233
elseif, 233
function, 233
i, 233
if, 233
receivedFormat, 234
return, 234
while, 233

EpuckPort
  CloseEpuck.m, 231
  EpuckFlush.m, 232
  OpenEpuck.m, 236
EpuckSendData.m
  function, 235
  fwrite, 235
  if, 235
  int16, 235
  return, 235
  size, 235
ERROR
  e_I2C_master_module.h, 224
Error
  OpenEpuck.m, 236
EXIT_OK
  e_agenda.c, 242
  e_agenda_fast.c, 314
EXPOSURE_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
FALSE
  e_epuck_ports.h, 326
fast_2_timer/e_calc.c
  ARRAY_ORIGINE_X, 121
  ARRAY_ORIGINE_Y, 121
  e_po3030k_config_cam, 121
  e_po3030k_get_bytes_per_pixel, 122
  e_po3030k_init_cam, 122
  IRQ_PIX_LAT, 121
fast_2_timer/e_po3030k.h
  ARRAY_HEIGHT, 126
  ARRAY_WIDTH, 126
  e_po3030k_apply_timer_config, 127
  e_po3030k_config_cam, 128
  e_po3030k_get_bytes_per_pixel, 129
  e_po3030k_get_register, 129
  e_po3030k_init_cam, 129
  e_po3030k_is_img_ready, 129
  e_po3030k_launch_capture, 130
  e_po3030k_read_cam_registers, 130
  e_po3030k_set_adc_offset, 130
  e_po3030k_set_ae_speed, 130
  e_po3030k_set_awb_ae, 130
  e_po3030k_set_awb_ae_tol, 131
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMMA_BASE</td>
<td>160</td>
</tr>
<tr>
<td>GAMMASELCOL_BASE</td>
<td>160</td>
</tr>
<tr>
<td>INTEGŔ_BASE</td>
<td>160</td>
</tr>
<tr>
<td>LENSQ_BASE</td>
<td>160</td>
</tr>
<tr>
<td>MCLK</td>
<td>160</td>
</tr>
<tr>
<td>MCLK_P_NS</td>
<td>160</td>
</tr>
<tr>
<td>MINMAXAWB_BASE</td>
<td>160</td>
</tr>
<tr>
<td>MINMAXEXP_BASE</td>
<td>160</td>
</tr>
<tr>
<td>MIRROR_BASE</td>
<td>160</td>
</tr>
<tr>
<td>MODE_GRAYSCALE</td>
<td>160</td>
</tr>
<tr>
<td>MODE_R5G6B5</td>
<td>160</td>
</tr>
<tr>
<td>MODE_YUV</td>
<td>160</td>
</tr>
<tr>
<td>NB_REGISTERS</td>
<td>160</td>
</tr>
<tr>
<td>po3030k_get_pixelclock</td>
<td>172</td>
</tr>
<tr>
<td>REFREPO_BASE</td>
<td>160</td>
</tr>
<tr>
<td>SAMPLING_ADDR</td>
<td>160</td>
</tr>
<tr>
<td>SEPIA_BASE</td>
<td>160</td>
</tr>
<tr>
<td>SEPIATONE_BASE</td>
<td>160</td>
</tr>
<tr>
<td>SPEED_ADDR</td>
<td>160</td>
</tr>
<tr>
<td>VSYNCCOL_BASE</td>
<td>160</td>
</tr>
<tr>
<td>VSYNCSSTART_BASE</td>
<td>160</td>
</tr>
<tr>
<td>VSYNSCSTOP_BASE</td>
<td>160</td>
</tr>
<tr>
<td>WEIGHWIN_BASE</td>
<td>160</td>
</tr>
<tr>
<td>WINDOW_X1_BASE</td>
<td>160</td>
</tr>
<tr>
<td>WINDOW_X2_BASE</td>
<td>160</td>
</tr>
<tr>
<td>WINDOW_Y1_BASE</td>
<td>160</td>
</tr>
<tr>
<td>WINDOW_Y2_BASE</td>
<td>160</td>
</tr>
<tr>
<td>WW_BASE</td>
<td>160</td>
</tr>
<tr>
<td>fast_2_timer/e_timers.c</td>
<td>189</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td>189</td>
</tr>
<tr>
<td>_po3030k_buffer</td>
<td>191</td>
</tr>
<tr>
<td>_po3030k_current_row</td>
<td>191</td>
</tr>
<tr>
<td>_po3030k_img_ready</td>
<td>191</td>
</tr>
<tr>
<td>_po3030k_line_conf</td>
<td>191</td>
</tr>
<tr>
<td>_po3030k_row</td>
<td>191</td>
</tr>
<tr>
<td>blank_row_betw</td>
<td>191</td>
</tr>
<tr>
<td>e_po3030k_apply_timer_config</td>
<td>189</td>
</tr>
<tr>
<td>e_po3030k_is_img_ready</td>
<td>190</td>
</tr>
<tr>
<td>e_po3030k_line_conf</td>
<td>191</td>
</tr>
<tr>
<td>init_timer4</td>
<td>190</td>
</tr>
<tr>
<td>init_timer5</td>
<td>191</td>
</tr>
<tr>
<td>fclose</td>
<td>231</td>
</tr>
<tr>
<td>FCY</td>
<td>326</td>
</tr>
<tr>
<td>FFT</td>
<td>25</td>
</tr>
<tr>
<td>fft/ Directory Reference</td>
<td>46</td>
</tr>
<tr>
<td>fft/e_fft.c</td>
<td>214</td>
</tr>
<tr>
<td>fft/e_fft.h</td>
<td>215</td>
</tr>
<tr>
<td>fft/e_fft_utilities.h</td>
<td>216</td>
</tr>
<tr>
<td>fft/e_input_signal.c</td>
<td>217</td>
</tr>
<tr>
<td>fft/e_twiddle_factors.c</td>
<td>218</td>
</tr>
<tr>
<td>FFT_BLOCK_LENGTH</td>
<td>215</td>
</tr>
<tr>
<td>FILTER_SIZE</td>
<td>71</td>
</tr>
<tr>
<td>find_function</td>
<td>317</td>
</tr>
<tr>
<td>FIRSTDATA_IN_PACKET_OFFSET</td>
<td>211</td>
</tr>
<tr>
<td>FLICKM_BASE</td>
<td>160</td>
</tr>
<tr>
<td>慢_3_timer/e_registers.c</td>
<td>176</td>
</tr>
<tr>
<td>FLICKP_BASE</td>
<td>160</td>
</tr>
<tr>
<td>flow_led</td>
<td>255</td>
</tr>
<tr>
<td>flush</td>
<td>268</td>
</tr>
<tr>
<td>flushinput</td>
<td>232</td>
</tr>
<tr>
<td>flushoutput</td>
<td>232</td>
</tr>
<tr>
<td>fopen</td>
<td>236</td>
</tr>
<tr>
<td>FOSC</td>
<td>326</td>
</tr>
<tr>
<td>FRAME_HEIGHT</td>
<td>160</td>
</tr>
<tr>
<td>FRAME_WIDTH</td>
<td>160</td>
</tr>
<tr>
<td>friendly_name</td>
<td>59</td>
</tr>
<tr>
<td>FRONT_LED</td>
<td>326</td>
</tr>
<tr>
<td>FRONT_LED_DIR</td>
<td>326</td>
</tr>
<tr>
<td>function</td>
<td>57</td>
</tr>
<tr>
<td>fwrite</td>
<td>232</td>
</tr>
<tr>
<td>GAMMA_BASE</td>
<td>160</td>
</tr>
<tr>
<td>slow_3_timer/e_registers.c</td>
<td>176</td>
</tr>
<tr>
<td>GAMMASELCOL_BASE</td>
<td>160</td>
</tr>
<tr>
<td>slow_3_timer/e_registers.c</td>
<td>176</td>
</tr>
<tr>
<td>GetHardwareAttenuator</td>
<td>211</td>
</tr>
<tr>
<td>ComModule.c</td>
<td>213</td>
</tr>
<tr>
<td>ComModule.c</td>
<td>211</td>
</tr>
</tbody>
</table>

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
GetTypeOwnAddress
    ComModule.c, 211
GetOwnGroup
    ComModule.c, 211
    ComModule.h, 213
GetRadioEnabledState
    ComModule.c, 211
    ComModule.h, 213
GetSoftwareAttenuator
    ComModule.c, 211
    ComModule.h, 213
GetStatus
    ComModule.c, 211
    ComModule.h, 213
goal_active_left
    e_motors_kml.c, 305
goal_active_right
    e_motors_kml.c, 305
goal_steps_left
    e_motors_kml.c, 305
goal_steps_right
    e_motors_kml.c, 305
GREY_SCALE_MODE
    fast_2_timer/e_po3030k.h, 126
    slow_3_timer/e_po3030k.h, 142
GROUPDATA_IN_PACKET_OFFSET
    ComModule.c, 211
HARDWAREATT_SET_FLAG
    ComModule.c, 211
if
    EpuckGetData.m, 233
    EpuckSendData.m, 235
img_ready
    slow_3_timer/e_timers.c, 194
inclination
    e_acc.c, 68
    TypeAccSpheric, 62
init_timer1
    slow_3_timer/e_timers.c, 193
init_timer4
    fast_2_timer/e_timers.c, 190
    slow_3_timer/e_timers.c, 194
init_timer5
    fast_2_timer/e_timers.c, 191
    slow_3_timer/e_timers.c, 194
init_tmr1
    e_proxy.c, 95
init_tmr2
    e_proxy_timer2.c, 108
init_value_ir
    advance_ad_scan/e_proxy.c, 92
InitComModule
    ComModule.c, 211
    ComModule.h, 213
INPUT_PIN
    e_epuck_ports.h, 326
int16
    EpuckSendData.m, 235
INTEGR_BASE
    fast_2_timer/e_registers.c, 160
    slow_3_timer/e_registers.c, 176
INTERRUPT_DELAY
    e_epuck_ports.h, 326
INTERRUPT_OFF
    e_epuck_ports.h, 326
INTERRUPT_ON
    e_epuck_ports.h, 326
IR0
    e_epuck_ports.h, 326
IR1
    e_epuck_ports.h, 326
IR2
    e_epuck_ports.h, 326
IR3
    e_epuck_ports.h, 326
IR4
    e_epuck_ports.h, 326
IR5
    e_epuck_ports.h, 326
IR6
    e_epuck_ports.h, 326
IR7
    e_epuck_ports.h, 326
e_epuck_ports.h, 326

IRQ_PIX_LAT
    fast_2_timer/e_calc.c, 121
is_ad_acquisition_completed
    advance_ad_scan/e_ad_conv.c, 76
is_ad_array_filled
    advance_ad_scan/e_ad_conv.c, 76
IsModulePlugged
    ComModule.c, 211
    ComModule.h, 213
IsPacketReady
    ComModule.c, 211
    ComModule.h, 213

k2000_led
    advance_one_timer/eLed.c, 255
    advance_one_timer/eLed.h, 268

LED0
    e_epuck_ports.h, 326

LED0_DIR
    e_epuck_ports.h, 326

LED1
    e_epuck_ports.h, 326

LED1_DIR
    e_epuck_ports.h, 326

LED2
    e_epuck_ports.h, 326

LED2_DIR
    e_epuck_ports.h, 326

LED3
    e_epuck_ports.h, 326

LED3_DIR
    e_epuck_ports.h, 326

LED4
    e_epuck_ports.h, 326

LED4_DIR
    e_epuck_ports.h, 326

LED5
    e_epuck_ports.h, 326

LED5_DIR
    e_epuck_ports.h, 326

LED6
    e_epuck_ports.h, 326

LED6_DIR
    e_epuck_ports.h, 326

LED7
    e_epuck_ports.h, 326

LED7_DIR
    e_epuck_ports.h, 326

LED_EFFECTS
    advance_one_timer/eLed.c, 252
left_led
    advance_one_timer/eLed.c, 255

left_motors
    advance_one_timer/e_motors.c, 279
    advance_one_timer/fast_agenda/e_motors.c, 283
    e_motors.kml.c, 305

left_speed
    advance_one_timer/e_motors.c, 279
    advance_one_timer/fast_agenda/e_motors.c, 283
    e_motors.c, 286
    e_motors.kml.c, 305
    e_motors_timer3.c, 332

LENSG_BASE
    fast_2_timer/e_registers.c, 160
    slow_3_timer/e_registers.c, 176

LIS sensor turret, 23
local_bt_PIN
    e_bluetooth.c, 114

LOG2_BLOCK_LENGTH
    e_fft.h, 215

Matlab communication, 31
matlab.c
    e_receive_char_from_matlab, 237
    e_receive_int_from_matlab, 237
    e_send_char_to_matlab, 238
    e_send_int_to_matlab, 238

matlab.h
    e_receive_char_from_matlab, 239
    e_receive_int_from_matlab, 239
    e_send_char_to_matlab, 240
    e_send_int_to_matlab, 240

matlab/ Directory Reference, 49
matlab/matlab files/ Directory Reference, 50
matlab/matlab files/CloseEpuck.m, 231
matlab/matlab files/EpuckFlush.m, 232
matlab/matlab files/EpuckGetData.m, 233
matlab/matlab files/EpuckSendData.m, 235
matlab/matlab files/OpenEpuck.m, 236
matlab/matlab.c, 237
matlab/matlab.h, 239

max_col
    slow_3_timer/e_timers.c, 194
max_row
    slow_3_timer/e_timers.c, 195

MAXV
    advance_one_timer/e_motors.c, 277
    advance_one_timer/fast_agenda/e_motors.c, 281
    e_motors.kml.c, 302

MCLK
    fast_2_timer/e_registers.c, 160
    slow_3_timer/e_registers.c, 176
MCLK_P_NS
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MCLK buffer
  advance_ad_scan/e_registers.c, 160
MIC0_BUFFER
  advance_ad_scan/e_registers.c, 160
MIC1
  e_epuck_ports.h, 326
MIC1 BUFFER
  advance_ad_scan/e_registers.c, 160
MIC2
  e_epuck_ports.h, 326
MIC2 BUFFER
  advance_ad_scan/e_registers.c, 160
MIC3
  e_epuck_ports.h, 326
MIC SAMP_FREQ
  advance_ad_scan/e_ad_conv.c, 79
MIC SAMP NB
  advance_ad_scan/e_ad_conv.c, 79
MICRO ONLY
  advance_ad_scan/e_ad_conv.c, 79
micro only
  advance_ad_scan/e_ad_conv.c, 76
MICROSEC
  e_epuck_ports.h, 326
migrate
  e_agenda_fast.c, 317
MILLISEC
  e_epuck_ports.h, 326
MINMAXAWB_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MINMAXEXP_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MIRROR_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MODE_GRAYSCALE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MODE_QQVGA
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MODE_QVGA
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MODE_R5G6B5
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MODE_VGA
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MODE_YUV
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
MOTOR1 PHA
  e_epuck_ports.h, 326
MOTOR1 PHA DIR
  e_epuck_ports.h, 326
MOTOR1 PHB
  e_epuck_ports.h, 326
MOTOR1 PHB DIR
  e_epuck_ports.h, 326
MOTOR1 PHC
  e_epuck_ports.h, 326
MOTOR1 PHC DIR
  e_epuck_ports.h, 326
MOTOR1 PHD
  e_epuck_ports.h, 326
MOTOR1 PHD DIR
  e_epuck_ports.h, 326
MOTOR2 PHA
  e_epuck_ports.h, 326
MOTOR2 PHA DIR
  e_epuck_ports.h, 326
MOTOR2 PHB
  e_epuck_ports.h, 326
MOTOR2 PHB DIR
  e_epuck_ports.h, 326
MOTOR2 PHC
  e_epuck_ports.h, 326
MOTOR2 PHC DIR
  e_epuck_ports.h, 326
MOTOR2 PHD
  e_epuck_ports.h, 326
MOTOR2 PHD DIR
  e_epuck_ports.h, 326
motor_counter_left
  e_motors_timer3.c, 332
motor_counter_left_init
  e_motors_timer3.c, 332
motor_counter_right
  e_motors_timer3.c, 332
motor_counter_right_init
  e_motors_timer3.c, 332
motor led/ Directory Reference
  motorLed/e_agenda.c, 241
motor led/ Directory Reference
  motorLed/e_led.c, 250
motor led/ Directory Reference
  motorLed/e_motors.c, 276
motor led/ Directory Reference
  motorLed/e_motors_kml.c, 301

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
INDEX

motor_led/advance_one_timer/e_remote_control.c, 306
motor_led/advance_one_timer/e_remote_control.h, 309
motor_led/advance_one_timer/fast_agenda/Directory Reference, 45
motor_led/advance_one_timer/fast_agenda/e_agenda_fast.c, 313
motorLed/advance_one_timer/fast_agenda/e_agenda_fast.h, 318
motorLed/advance_one_timer/fast_agenda/e_led.c, 256
motorLed/advance_one_timer/fast_agenda/e_led.h, 269
motorLed/advance_one_timer/fast_agenda/e_motors.c, 280
motorLed/advance_one_timer/fast_agenda/e_motors.h, 292
motor Led/e_epuck_ports.h, 323
motor Led/e_init_port.c, 327
motor Led/e_init_port.h, 328
motor Led/e_led.c, 261
motor Led/e_led.h, 274
motor Led/e_motors.c, 284
motor Led/e_motors.h, 297
motor Led/e_motors_timer3.c, 329
motors
AgendaList, 55
MUTE
  e_remote_control.h, 311
my ceil
  e_agenda_fast.c, 317
NANOSEC
  e_epuck_ports.h, 326
NB_REGISTERS
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
nbr_pas_left
  e_motors.c, 286
  e_motors_timer3.c, 332
nbr_pas_right
  e_motors.c, 286
  e_motors_timer3.c, 332
nbr_steps_left
  advance_one_timer/e_motors.c, 279
  advance_one_timer/fast_agenda/e_motors.c, 283
  e_motors_kml.c, 305
nbr_steps_right
  advance_one_timer/e_motors.c, 279
  advance_one_timer/fast_agenda/e_motors.c, 283
  e_motors_kml.c, 305
next
  AgendaType, 57, 58
NOP
  e_epuck_ports.h, 326
number
  BtEPuck, 60
OpenEpuck.m
  EpuckPort, 236
  Error, 236
  fopen, 236
  port, 236
  return, 236
OPERATION_OK
  e_I2C_master_module.h, 224
orientation
  e_acc.c, 68
  TypeAccSpheric, 62
OUT_AUX_1
  e_remote_control.h, 311
OUTPUT_PIN
  e_epuck_ports.h, 326
OWNADDRH_REG_ADDR
  ComModule.c, 211
OWNADDRL_REG_ADDR
  ComModule.c, 211
OWNGROUP_REG_ADDR
  ComModule.c, 211
PACKET_LOST_FLAG
  ComModule.c, 211
PACKET_READY_FLAG
  ComModule.c, 211
pbp_current
  slow_3_timer/e_timers.c, 195
pixel_betw_pixel
  slow_3_timer/e_timers.c, 195
PLL
  e_epuck_ports.h, 326
PO3030K_FULL
  fast_2_timer/e_po3030k.h, 126
  slow_3_timer/e_po3030k.h, 142
po3030k_get_pixelclock
  fast_2_timer/e_registers.c, 172
  slow_3_timer/e_registers.c, 188
port
  OpenEpuck.m, 236
Ports, motors and LEDs, 32
POWERSAVE
  advance_one_timer/e_motors.c, 277
  advance_one_timer/fast_agenda/e_motors.c, 281
  e_motors_kml.c, 302
PULSE_IR0
Radio communication, 24
RADIO_ENABLED_FLAG
ComModule.c, 211
READ
  e_i2c_master_module.h, 224
ReadRegister
  ComModule.c, 211
REC_BUFFER_END
  ComModule.c, 211
REC_BUFFER_START
  ComModule.c, 211
receivedFormat
  EpuckGetData.m, 234
recompute_speeds
  e_agenda_fast.c, 317
reflected_ir
  e_prox.c, 95
e_prox_timer2.c, 108
REFREPO_BASE
  fast_2_timer/e_registers.c, 160
slow_3_timer/e_registers.c, 176
REMOTE
  e_epuck_ports.h, 326
REMOTE_DIR
  e_epuck_ports.h, 326
REQUEST_TO_SEND_FLAG
  ComModule.c, 211
RESET
  e_epuck_ports.h, 326
RESTART
  e_i2c_master_module.h, 224
return
  EpuckGetData.m, 234
  EpuckSendData.m, 235
INDEX

e_po3030k_set_sepia, 153
e_po3030k_set_sepia_tone, 153
e_po3030k_set_speed, 153
e_po3030k_set_vsync, 154
e_po3030k_set_weight_win, 154
e_po3030k_set_wx, 155
e_po3030k_set_wy, 155
e_po3030k_sync_register_array, 156
e_po3030k_write_cam_registers, 156
e_po3030k_write_gamma_coef, 156
GREY_SCALE_MODE, 142
MODE_QQVGA, 142
MODE_QVGA, 142
MODE_VGA, 142
PO3030K_FULL, 142
RGB_565_MODE, 142
SPEED_128, 143
SPEED_16, 143
SPEED_2, 143
SPEED_2_3, 143
SPEED_32, 143
SPEED_4, 143
SPEED_64, 143
SPEED_8, 143
YUV_MODE, 143

slow_3_timer/e_registers.c

ADCOFF_BASE, 176
AESPEED_BASE, 176
ARRAY_ORIGINE_X, 176
ARRAY_ORIGINE_Y, 176
AWBAEENABLE_BASE, 176
AWVAETOL_BASE, 176
BASE_D1, 176
BASE_D2, 176
BASE_D3, 176
BASE_D4, 176
BIAS_BASE, 176
BRICTR_BASE, 176
cam_reg, 188
CBCRGAIN_BASE, 176
COLGAIN_BASE, 176
COLOR_COEF_BASE, 176
COLOR_M_ADDR, 176
DEVICE_ID, 176
e_po3030k_get_register, 176
e_po3030k_read_cam_registers, 177
e_po3030k_set_adc_offset, 177
e_po3030k_set_ac_speed, 177
e_po3030k_set_awb_ac, 177
e_po3030k_set_awb_ac_tol, 178
e_po3030k_set_bias, 178
e_po3030k_set_brigh_contr, 178
e_po3030k_set_cb_cr_gain, 178
e_po3030k_set_color_gain, 179
e_po3030k_set_color_mode, 179
e_po3030k_set_edge_prop, 179
e_po3030k_set_exposure, 180
e_po3030k_set_flicker_detection, 180
e_po3030k_set_flicker_mode, 181
e_po3030k_set_flicker_man_set, 180
e_po3030k_setgamma_coef, 181
e_po3030k_set_integr_time, 181
e_po3030k_set_lens_gain, 182
e_po3030k_set_max_min_awb, 182
e_po3030k_set_max_min_exp, 182
e_po3030k_set_mirror, 183
e_po3030k_set_ref_exposure, 183
e_po3030k_set_register, 183
e_po3030k_set_sampling_exposure, 184
e_po3030k_set_sepia_mode, 184
e_po3030k_set_sepia, 184
e_po3030k_set_sepia_tone, 184
e_po3030k_set_speed, 184
e_po3030k_set_vsync, 185
e_po3030k_set_weight_win, 185
e_po3030k_set_wx, 186
e_po3030k_set_wy, 186
e_po3030k_set_max_min_awb, 182
e_po3030k_set_max_min_exp, 182
e_po3030k_set_mirror, 183
e_po3030k_set_register, 183
e_po3030k_write_cam_registers, 187
e_po3030k_write_gamma_coef, 187

EDGE_BASE, 176
EXPOSURE_BASE, 176
FLICKM_BASE, 176
FLICKP_BASE, 176
FRAME_HEIGHT, 176
FRAME_WIDTH, 176
GAMMA_BASE, 176
GAMMASELCOL_BASE, 176
INTEGR_BASE, 176
LENSG_BASE, 176
MCLK, 176
MCLK_P_NS, 176
MINMAXAWB_BASE, 176
MINMAXEXP_BASE, 176
MIRROR_BASE, 176
MODE_GRAYSCALE, 176
MODE_R5G6B5, 176
MODE_YUV, 176
NB_REGISTERS, 176
po3030k_read_pixelclock, 188
REFREPO_BASE, 176
SAMPLING_ADDR, 176
SEPIA_BASE, 176
SEPIATONE_BASE, 176
SPEED_ADDR, 176
VSYNCCOL_BASE, 176

Generated on Fri Nov 9 06:32:57 2007 for e-puck by Doxygen
VSYNCSTART_BASE, 176
VSYNCSTOP_BASE, 176
WEIGHWIN_BASE, 176
WINDOW_X1_BASE, 176
WINDOW_X2_BASE, 176
WINDOW_Y1_BASE, 176
WINDOW_Y2_BASE, 176
WW_BASE, 176
slow_3_timer/e_timers.c
  __attribute__, 192
  bbl_current, 194
  blank_betw_lines, 194
  bpp_current, 194
  buf_pos, 194
  buffer, 194
  bytes_per_pixel, 194
  current_col, 194
  current_row, 194
  e_po3030k_apply_timer_config, 193
  e_po3030k_is_img_ready, 193
  e_po3030k_launch_capture, 193
  img_ready, 194
  init_timer1, 193
  init_timer4, 194
  init_timer5, 194
  max_col, 194
  max_row, 195
  pbp_current, 195
  pixel_betw_pixel, 195
snake_led
  advance_one_timer/e_led.c, 255
  advance_one_timer/e_led.h, 268
SOFTATT_REG_ADDR
  ComModule.c, 211
Sound, 21
speed
  AgendaList, 56
SPEED_128
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_16
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_2
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_2_3
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_32
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_4
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
  SPEED_64
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_8
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143
SPEED_ADDR
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
STANDBY
  e_remote_control.h, 311
START
  e_I2C_master_module.h, 224
STATUS_REG_ADDR
  ComModule.c, 211
STOP
  e_I2C_master_module.h, 224
STOP_TMR1
  e_epuck_ports.h, 326
STOP_TMR2
  e_epuck_ports.h, 326
STOP_TMR3
  e_epuck_ports.h, 326
STOP_TMR4
  e_epuck_ports.h, 326
STOP_TMR5
  e_epuck_ports.h, 326
TCY_PIC
  e_epuck_ports.h, 326
timers_in_use
  AgendaList, 55
TRESHV
  advance_one_timer/e_motors.c, 277
  advance_one_timer/fast_agenda/e_motors.c, 281
  e_motors_kml.c, 302
TRUE
  e_epuck_ports.h, 326
TX_IDLE_FLAG
  ComModule.c, 211
TX_SEND_ERROR
  ComModule.c, 211
TypeAccRaw, 61
  acc_x, 61
  acc_y, 61
  acc_z, 61
TypeAccSpheric, 62
  acceleration, 62
  inclination, 62
  orientation, 62
TYPEDATA_IN_PACKET_OFFSET
  ComModule.c, 211
UART, 36
uart/ Directory Reference, 54
uart/e_epuck_ports.inc, 333
uart/e_uart_char.h, 334

VOL_DOWN
  e_remote_control.h, 311
VOL_UP
  e_remote_control.h, 311
VSYNC_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
VSYNC_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
VSYNCSTOP_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176

waiting
  AgendaList, 55
WEIGHWIN_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
while
  EpuckGetData.m, 233
WINDOW_X1_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
WINDOW_X2_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
WINDOW_Y1_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
WINDOW_Y2_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176
WRITE
  e_I2C_master_module.h, 224
WriteRegister
  ComModule.c, 211
WW_BASE
  fast_2_timer/e_registers.c, 160
  slow_3_timer/e_registers.c, 176

YUV_MODE
  fast_2_timer/e_po3030k.h, 127
  slow_3_timer/e_po3030k.h, 143