OpenSwarm

0.15.12.2

Generated by Doxygen 1.8.9.1

Sat Jan 9 2016 00:20:11

Contents

Chapter 1

OpenSwarm documentation

1.1 Introduction

OpenSwarm is an easy-to-use event-driven preemptive operating system for miniature robots. It offers abstract hardware-independent functions to make user code more extendible, maintainable, and portable. The hybrid kernel provides preemptive and cooperative scheduling, asynchronous programming models with events, and inter-process communication functions.

OpenSwarm was created during the PhD of Stefan M Trenkwalder (http://trenkwalder.tech) at the University of Sheffield (http://www.sheffield.ac.uk/) under the Supervision of Dr. Roderich Gross and Dr. Andreas Kolling.

The code of OpenSwarm can be basically divided into 3 different modules:

- Process Manages
- Event System
- I/O Management (This includes device specific sensors and actuators)

1.2 Documentation Structure

This documentation was generated by Doxygen (http://www.doxygen.org) and is structured as follows

- Main Page: This tab represents an short introduction to and general comments on OpenSwarm
- Modules: This tab presents a list of logical units of OpenSwarm (such as Process Management or Event System)
- Data Structures: This tab shows a list of all used data structures inside OpenSwarm.
- Files: This tab lists the documentation of each individual file in OpenSwarm.

1.3 Links

- http://www.openswarm.org/ The official OpenSwarm website
- http://trenkwalder.tech/ The academic webpage of Stefan Trenkwalder
- http://naturalrobotics.group.shef.ac.uk/ The website of the research group
- http://openswarm.org/license/ The link to the newest license (in case it changed)

1.4 License

1.5 Thanks

OpenSwarm is part of the PhD of Stefan M. Trenkwalder (http://trenkwalder.tech) who is recipient of a DOC Fellowship of the Austrian Academy of Sciences (http://www.oeaw.ac.at/).

Chapter 2

Todo List

Module camera

The used functions from the e-puck library are very time and computational intensive. These function can be rewritten to decrease the processing load.

File camera.c

The used functions from the e-puck library are very time and computational intensive. These function can be rewritten to decrease the processing load.

Module i2c

testing and debugging of this module.

globalScope> Global Sys_Camera_PreProcessor (void)

rewrite the camera to computational less intensive functions

globalScope> Global Sys_Init_Camera (void)

rewrite the camera to computational less intensive functions

globalScope> Global Sys_Init_Camera (void)

rewrite the camera to computational less intensive functions

globalScope> Global Sys_Start_Camera (void)

rewrite the camera to computational less intensive functions

globalScope> Global Sys_Start_Camera (void)

rewrite the camera to computational less intensive functions

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

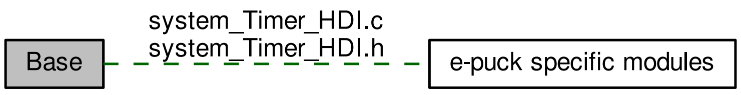
Chapter 6

Module Documentation

6.1 Base

Basic functions to start and initialise the operating system; allocate, free and copy memory, and create and end atomic sections.

Collaboration diagram for Base:



Files

- file interrupts.c

  includes basic system calls to create atomic sections. (Sections that cannot be interrupted)
- file interrupts.h

  includes system calls to create atomic sections. (Sections that cannot be interrupted)
- file memory.c

  includes functions to allocate, free, and copy memory
- file memory.h

  includes functions to allocate, free, and copy memory
- file system.c

  includes system calls that initialise and configure the operating system.
- file system.h

  initiaises and starts OpenSwarm.
- file system_Timer_HDI.c

  Hardware dependent implementations to initialise, configure and the operating system.
- file system_Timer_HDI.h

  Hardware dependent implementations to initialise, configure and the operating system.

6.1.1 Detailed Description

Basic functions to start and initialise the operating system; allocate, free and copy memory, and create and end atomic sections.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

6.1.2 Introduction

This package contains basic functions to initialise and start all modules of OpenSwarm. This part of OpenSwarm executes all three modules

1. Process Management

   See also

   procMan

   See also

2. Event System

3. I/O Management (This includes device specific sensors and actuators)

   See also

   IOMan

   It first defines global preprocessor option to configure OpenSwarm. It initialises the system and I/O according to its configuration (preprocessor definitions) and with an additional command the system can be started. In addition, functions to define atomic sections (sections that cannot be interrupted by anything), allocate and free memory are also provided.

6.1.2.1 Definitions

definition.h provides standardisation ports, configuration the used platform, and general preprocessor/type definitions that are needed in the entire OpenSwarm project.

6.1.2.2 Memory Management

OpenSwarm is designed for processing unit that lack a MMU (Memory Management Unit). As a consequence, advance memory management functions as virtual memory cannot be implemented without a significant reduction of efficiency. OpenSwarm provides atomic functions to allocate, free and copy memory in memory.h.

6.1.2.3 Interrupt Management

OpenSwarm provides a clear structure of interrupt priorities and functions to create atomic sections in interrupts.h.

6.1.2.4 Dependencies

This part of OpenSwarm executes all three modules and depends on the configuration of each part and its implementation:

- Process Management

See also

procMan

- Event System

  See also

  EventSys

- I/O Management (This includes device specific sensors and actuators of used platform)

  See also

  IOMan

6.2 Event System

Functions to emit, create, (un)subscribe, (un)register, and manage events.

Files

- file events.c

    This file includes all system calls needed to create, (un)subscribe, (un)register, and delete events and related handler.
- file events.h

    functions to create handle and configure events.

6.2.1 Detailed Description

Functions to emit, create, (un)subscribe, (un)register, and manage events.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Events are a main component of OpenSwarm. It can be used to synchronise and communicate with processes, to implement asynchronous programming model, and process incoming data/signals.

6.2.2 Usage

The event system doesn't need to be initialised. Any event is identified by an integer **eventID**. To use an event the following steps have to be taken:

1. An event (**eventID**) can be (un)registered by Sys_Register_Event(uint16 eventID) and Sys_Unregister_Event(uint16 eventID). When an event is registered, it means that an event (**eventID**) can occur and handled by OpenSwarm.
2. After the event was registered, processes can be subscribed to it with Sys_Subscribe_to_Event(uint16 eventID, uint16 pid, pEventHandlerFunction handler, pConditionFunction condition) and Sys_Unsubscribe_from_Event(uint16 eventID, uint16 pid). During the subscription, an event handler (i.e. a function to process data that was sent by events) is subscribed to a specific event (**eventID**) and a process. Each event handler of a process for an specific event is unique. As a result the same handler function can be assigned to the same event if they are assigned to other processes.
3. After an event is registered, events can be sent with Sys_Send_Event(uint16 eventID, void *data, uint16 data_size) and Sys_Send_IntEvent(uint16 eventID, uint16 data).

6.2.3 Example

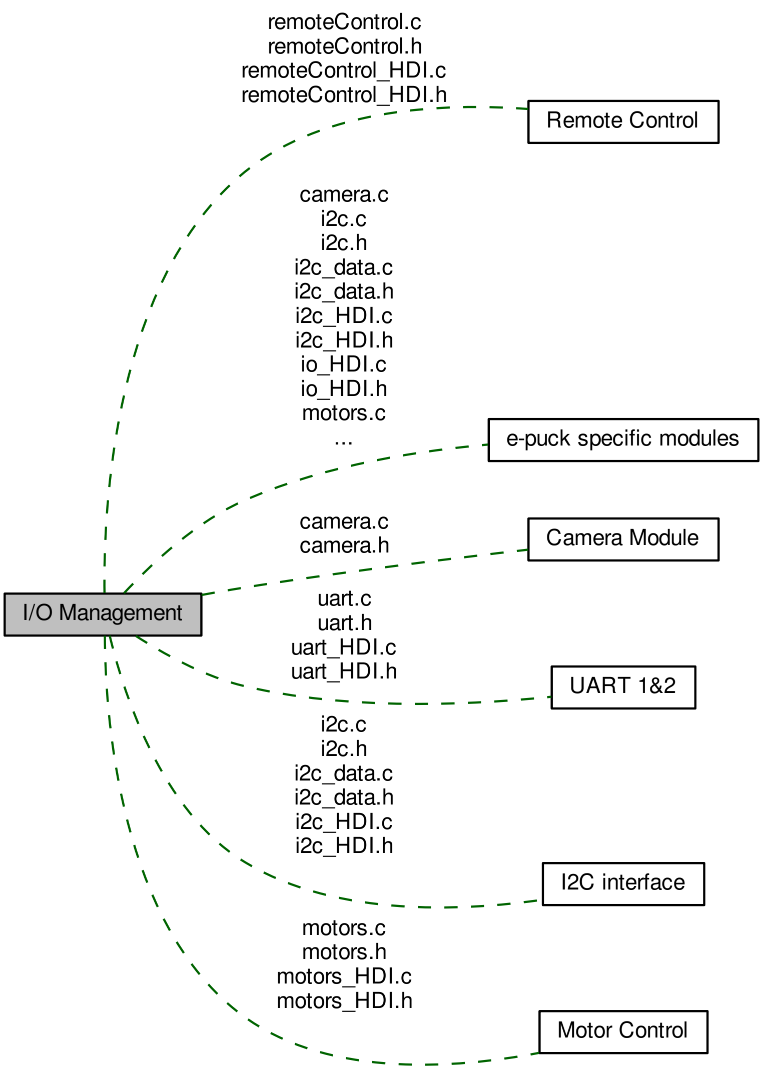
```
#include "os/system.h"
#include "os/events/events.h"

#define USER_EVENT_ID 0xCC

bool pConditionFunction(void *data){//only execute the the eventHandler every 5th time.
     static int counter = 0;

     if(++counter >= 4){//if event occurred 5 times
         counter = 0;
         return true;//execute eventHandler
     }

    return false;//don't execute eventHandler
}

bool eventHandler(uint16 pid, uint16 eventID, sys_event_data *data){
     //do something with the data
}

int main(void){
 //initialise some global or local variables

 int variable;

    Sys_Init_Kernel();

 Sys_Register_Event(USER_EVENT_ID);

 Sys_Start_Kernel();
    while(1){

     if( condition ){
          Sys_Send_Event(USER_EVENT_ID, &variable, sizeof(int));
     }
     //do something
    }
}
```

6.2.4 License

6.3 I/O Management

Functions and mechanisms to use I/O devices (e.g. sensors and actuators) to interact with the environment.

Collaboration diagram for I/O Management:



Files

- file io.c

  This file includes the IO timer to start and stop the timer. This timer executes IO functions periodically.
- file io.h

  This file includes the IO timer to start and stop the timer. This timer executes IO functions periodically.
- file io_clock.c

  This file includes the system clock that can be used to measure time.
- file io_clock.h

  This file includes the system clock that can be used to measure time.

- file camera.c

    This file includes functions to process data retrieved by a camera.
- file camera.h

    This file includes functions to process data retrieved by a camera.
- file i2c.c

    This file includes functions to read and write on the I2C interface.
- file i2c.h

    This file includes functions to read and write on the I2C interface.
- file i2c_data.c

    This file includes functions to read and write on the I2C interface.
- file i2c_data.h

    This file includes functions to read and write on the I2C interface.
- file i2c_HDI.c

    Hardware dependent implementations to read and write on the I2C interface.
- file i2c_HDI.h

    Hardware dependent implementations to read and write on the I2C interface.
- file io_HDI.c

    Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.
- file io_HDI.h

    Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.
- file motors.c

    This file provides the function needed to actuate the motors.
- file motors.h

    This file provides the function needed to actuate the motors.
- file motors_HDI.c

    Hardware dependent implementations to actuate the motors.
- file motors_HDI.h

    Hardware dependent implementations to actuate the motors.
- file remoteControl.c

    This file includes functions needed to receive and decode messages from a remote control.
- file remoteControl.h

    This file includes functions needed to receive and decode messages from a remote control.
- file remoteControl_HDI.c

    Hardware dependent implementations to receive and decode messages from a remote control.
- file remoteControl_HDI.h

    Hardware dependent implementations to receive and decode messages from a remote control.
- file uart.c

    This file includes functions needed to transmit data via uart(1 & 2).
- file uart.h

    This file includes functions needed to transmit data via uart(1 & 2).
- file uart_HDI.c

    Hardware dependent implementations to control the message flow of the UART interface.
- file uart_HDI.h

    Hardware dependent implementations to control the message flow of the UART interface.

6.3.1 Detailed Description

Functions and mechanisms to use I/O devices (e.g. sensors and actuators) to interact with the environment.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

6.3.2 Introduction

I/O device are managed by this module. I/O devices interfacing and interacting with the environment of the robot. These sensors and actuators might be a camera, motors, or gripper.

In general I/O devices might be independent and uses their own interrupts - such as UART, ADC, I2C. These functions act independently and only need to be initialised and started. No further interaction is needed.

Many I/O devices however need periodic interactions - such as remote control receiver, motor controller, or system clock.

6.3.3 Usage

The I/O management is initialised with Sys_Init_IOManagement(void), which initialised the System Timer (100us) and initialises a list of I/O devices that need to be executed periodically. After starting the timer with Sys_Start_I←OManagement(void), it can be the stopped with Sys_Stop_IOManagement(void).

The I/O Timer can be manipulated as follows

- Stop: Sys_Stop_IOTimer(void)
- Continue: Sys_Continue_IOTimer(void)
- Reset (starts the 100us again): Sys_Reset_IOTimer(void)
- Disable: Sys_Disable_IOTimerInterrupt(void)
- Enable: Sys_Enable_IOTimerInterrupt(void)
- Force an I/O Timer interrupt: Sys_Force_IOTimerInterrupt(void)

New I/O devices can be added and removed by (un)registering with Sys_Register_IOHandler(pFunction func) and Sys_Unregister_IOHandler(pFunction func).

The I/O management is started by initialising & starting of the kernel

See also

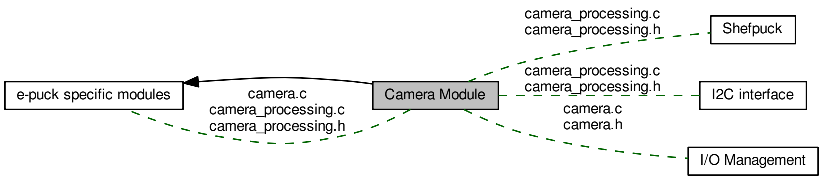
Base

6.3.4 License

6.4 Camera Module

Functions to process incoming frames from a camera module.

Collaboration diagram for Camera Module:



Files

- file camera.c

    This file includes functions to process data retrieved by a camera.
- file camera.h

    This file includes functions to process data retrieved by a camera.
- file camera_processing.c
- file camera_processing.h

6.4.1 Detailed Description

Functions to process incoming frames from a camera module.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

6.4.2 Introduction

This module is part of the I/O handler.

See also

I/O Management

This module currently is under development and is using functions of the e-puck library provided using Subversion at svn://svn.gna.org/svn/e-puck/trunk .

Todo The used functions from the e-puck library are very time and computational intensive. These function can be rewritten to decrease the processing load.

6.4.3 Usage

The camera is initialised and started by Sys_Init_Camera and Sys_Start_Camera respectively.

The camera uses a preprocessor to process a frame and generate the required events. This preprocessor can be defined by Sys_Set_Preprocessing(pCameraPreProcessor).

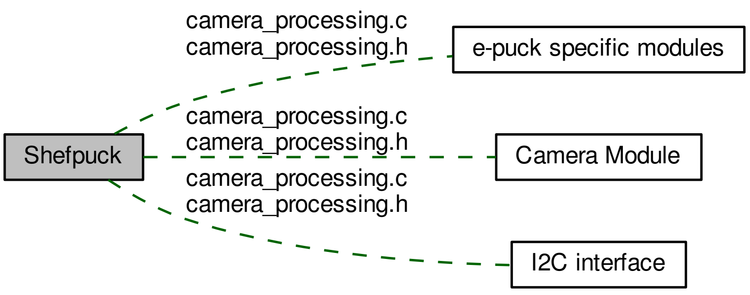
A received frame, if available (isNewFrameAvailable()) can be obtained with getFinishedFrame().

6.4.4 License

6.5 Shefpuck

External set of functions to assist the programming of the e-Puck.

Collaboration diagram for Shefpuck:



Files

- file camera_processing.c
- file camera_processing.h

6.5.1 Detailed Description

External set of functions to assist the programming of the e-Puck.

Author

Yuri Kaszubowski Lopes yurikazuba@gmail.com

This file is part of shefpuck.

This library is in development.

6.5.2 License

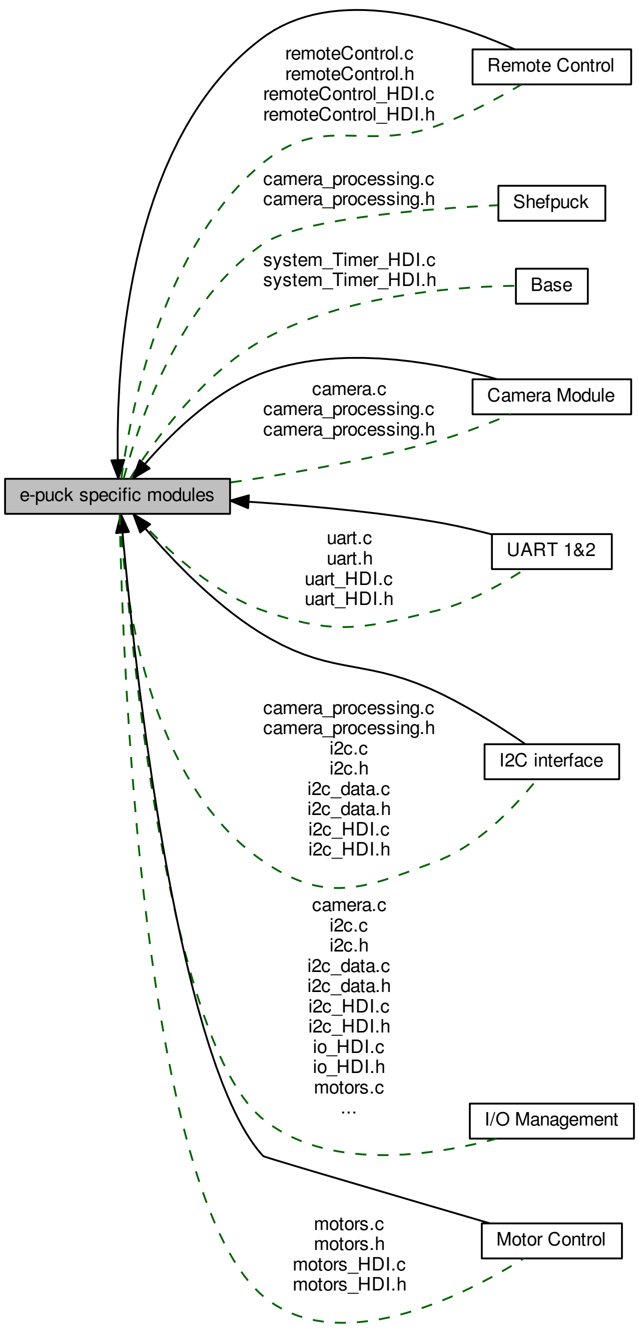
Note

This module is used because the e-puck library functions are used to access the camera. This module as well as the e-puck camera well be replaced.

6.6 e-puck specific modules

Modules and functions that are needed to use the e-puck platform.

Collaboration diagram for e-puck specific modules:



Modules

- Camera Module

  Functions to process incoming frames from a camera module.

- I2C interface

  Functions to read from and write on the I2C interface.
- Motor Control

  Functions to control the motors.
- Remote Control

  Functions to receive data from a remote control.
- UART 1&2

  Functions to control the message flow of the UART interface.

Files

- file camera.c

  This file includes functions to process data retrieved by a camera.
- file camera_processing.c
- file camera_processing.h
- file DSPIC30F6014A_HDI.h

  Hardware dependent declarations and definitions.
- file i2c.c

  This file includes functions to read and write on the I2C interface.
- file i2c.h

  This file includes functions to read and write on the I2C interface.
- file i2c_data.c

  This file includes functions to read and write on the I2C interface.
- file i2c_data.h

  This file includes functions to read and write on the I2C interface.
- file i2c_HDI.c

  Hardware dependent implementations to read and write on the I2C interface.
- file i2c_HDI.h

  Hardware dependent implementations to read and write on the I2C interface.
- file io_HDI.c

  Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.
- file io_HDI.h

  Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.
- file motors.c

  This file provides the function needed to actuate the motors.
- file motors.h

  This file provides the function needed to actuate the motors.
- file motors_HDI.c

  Hardware dependent implementations to actuate the motors.
- file motors_HDI.h

  Hardware dependent implementations to actuate the motors.
- file remoteControl.c

  This file includes functions needed to receive and decode messages from a remote control.
- file remoteControl.h

  This file includes functions needed to receive and decode messages from a remote control.
- file remoteControl_HDI.c

  Hardware dependent implementations to receive and decode messages from a remote control.
- file remoteControl_HDI.h

  Hardware dependent implementations to receive and decode messages from a remote control.
- file system_Timer_HDI.c

Hardware dependent implementations to initialise, configure and the operating system.
- file system_Timer_HDI.h

    Hardware dependent implementations to initialise, configure and the operating system.
- file traps.c

    Hardware dependent implementations to catch hardware traps.
- file uart.c

    This file includes functions needed to transmit data via uart(1 & 2).
- file uart.h

    This file includes functions needed to transmit data via uart(1 & 2).
- file uart_HDI.c

    Hardware dependent implementations to control the message flow of the UART interface.
- file uart_HDI.h

    Hardware dependent implementations to control the message flow of the UART interface.

6.6.1 Detailed Description

Modules and functions that are needed to use the e-puck platform.

See also

`http://www.gctronic.com/doc/index.php/E-Puck`

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

This module includes all other modules that are specific to the e-puck platform.

The e-puck provides the following features:

6.6.2 Sensors:

6.6.2.1 8 infra-red proximity sensors

The infra-red proximity sensors are currently under implementation. Therefore not ready yet.

6.6.2.2 accelerometer

The accelerometer weren't needed for many applications and, therefore, the priority to implement the speakers is small.

6.6.2.3 3 microphones

The microphones weren't needed for many applications and, therefore, the priority to implement the speakers is small.

6.6.2.4 camera:

The camera functions can be found at

See also

Camera Module

6.6.2.5 remote control receiver:

This function is fully implemented.

See also

Remote Control

6.6.3 Actuators:

6.6.3.1 differential drive \sa motors

6.6.3.2 leds:

Hardware independent functions to control the LEDs are not yet implemented, due to it's simple nature. Currently you can use the MACROs LED0, LED1, ..., LED7, BODYLED, FRONTLED to use the LEDs.

6.6.3.3 speaker:

The speakers weren't needed for many applications and, therefore, the priority to implement the speakers is small.

6.6.4 communication:

6.6.4.1 Bluetooth:

The Bluetooth can be used by sending and receiving bytes via UART1

See also

UART 1&2

6.6.4.2 Infra-red communication

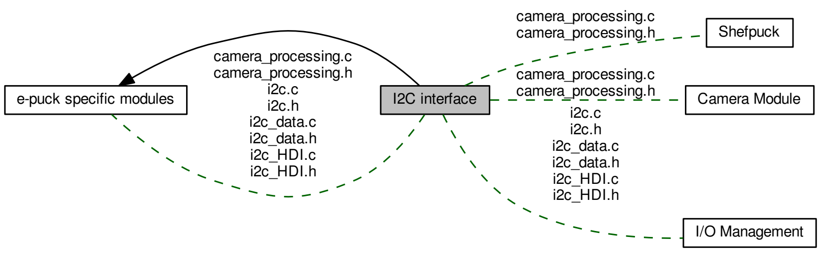
The infra-red proximity sensors can be used to transmit and receive data. This function leads to a local broadcasting. However, this function has not been implemented yet.

6.6.5 License

6.7 I2C interface

Functions to read from and write on the I2C interface.

Collaboration diagram for I2C interface:



Files

- file camera_processing.c
- file camera_processing.h
- file i2c.c

  This file includes functions to read and write on the I2C interface.
- file i2c.h

  This file includes functions to read and write on the I2C interface.
- file i2c_data.c

  This file includes functions to read and write on the I2C interface.
- file i2c_data.h

  This file includes functions to read and write on the I2C interface.
- file i2c_HDI.c

  Hardware dependent implementations to read and write on the I2C interface.
- file i2c_HDI.h

  Hardware dependent implementations to read and write on the I2C interface.

6.7.1 Detailed Description

Functions to read from and write on the I2C interface.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Inter-Integrated Circuit bus is a multi-master, multi-slave, serial bus (see also https://en.wikipedia.↩ org/wiki/I%C2%B2C)

OpenSwarm organises processes in three lists of processes (pid sorted):

1. running list: includes all processes are ready to be executed and are scheduled according to the scheduling algorithm.
2. blocked list: includes all processes that are waiting for events to occur.
3. Zombie list: includes all processes that are about to be terminated but not deleted yet.

6.7.2 Usage

THe I2C interface can be initialised and started with Sys_Init_I2C and Sys_Start_I2C respectively. Similarly, it can be paused, continued, or stopped by Sys_Pause_I2C, Sys_Contine_I2C, or Sys_Stop_I2C respectively. While the interface is running, data can be written with Sys_I2C_SentBytes. Values can be read with Sys_I2C_Read where the request message has also to be specified.

Todo testing and debugging of this module.

Note

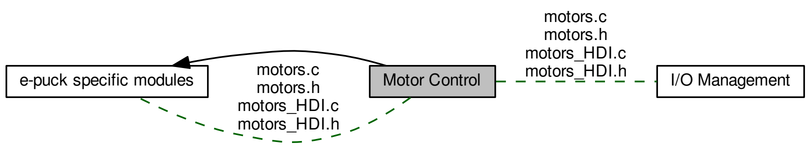
This module is currently untested. Might doesn't work or includes some bugs. The interrupt handler _M↩ I2CInterrupt is also out commented, because it might interfere with the e-Puck library used in the camera module.

6.7.3 License

6.8 Motor Control

Functions to control the motors.

Collaboration diagram for Motor Control:



Files

- file motors.c

  This file provides the function needed to actuate the motors.
- file motors.h

  This file provides the function needed to actuate the motors.
- file motors_HDI.c

  Hardware dependent implementations to actuate the motors.
- file motors_HDI.h

  Hardware dependent implementations to actuate the motors.

6.8.1 Detailed Description

Functions to control the motors.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

The motor control module controls the speed and motion of motors

6.8.2 Usage

After the initialisation with Sys_Init_Motors(), the motors can be used by setting the motor speed. This can be done by sending the motor velocities via events to SYS_EVENT_IO_MOTOR_LEFT and SYS_EVENT_IO_MOTOR←_RIGHT or by setting the speed directly by calling Sys_Set_LeftWheelSpeed(sint16) and Sys_Set_RightWheel←Speed(sint16). The current speed can be obtained Sys_get_LeftWheelSpeed() and Sys_get_RightWheelSpeed().

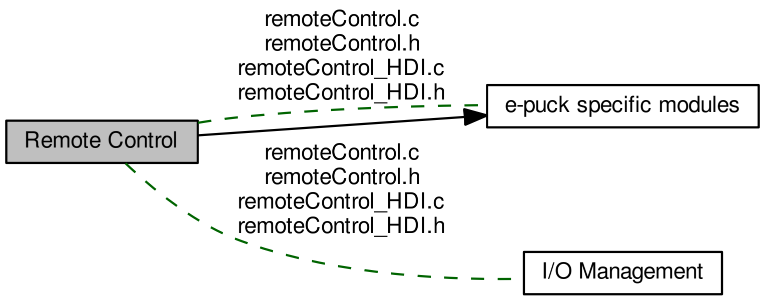
6.8.3 License

LICENSE: adapted FreeBSD License (see `http://openswarm.org/license`)
Copyright (c) 2015, Stefan M. Trenkwalder
All rights reserved.

6.9 Remote Control

Functions to receive data from a remote control.

Collaboration diagram for Remote Control:



Files

- file remoteControl.c

  This file includes functions needed to receive and decode messages from a remote control.
- file remoteControl.h

  This file includes functions needed to receive and decode messages from a remote control.
- file remoteControl_HDI.c

  Hardware dependent implementations to receive and decode messages from a remote control.
- file remoteControl_HDI.h

  Hardware dependent implementations to receive and decode messages from a remote control.

6.9.1 Detailed Description

Functions to receive data from a remote control.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

This module is based on the RC-5 coding for the (Toshiba RC-3910)

6.9.2 Usage

After the initialisation with Sys_Init_RemoteControl(), the interface needs to be started to be able to receive or transmit bytes with Sys_Start_RemoteControl().

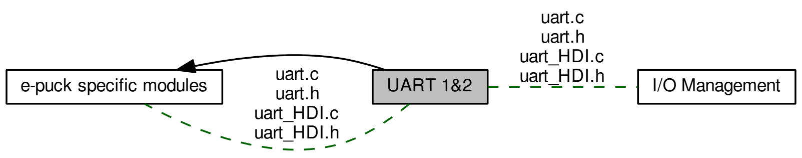
After this every button pressed on the remote control is received as an event (SYS_EVENT_IO_REMOECONTR↩ OL).

6.9.3 License

6.10 UART 1&2

Functions to control the message flow of the UART interface.

Collaboration diagram for UART 1&2:



Files

- file uart.c

  This file includes functions needed to transmit data via uart(1 & 2).
- file uart.h

  This file includes functions needed to transmit data via uart(1 & 2).
- file uart_HDI.c

  Hardware dependent implementations to control the message flow of the UART interface.
- file uart_HDI.h

  Hardware dependent implementations to control the message flow of the UART interface.

6.10.1 Detailed Description

Functions to control the message flow of the UART interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

A UART (Universal Asynchronous Receiver Transmitter) interface is common on microcontroller to communicate with other devices on a serial bus.

See also

`https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter`
The UART 1 is used on the e-puck specific modules to communicate with the Bluetooth transceiver.

6.10.2 Usage

After the initialisation with Sys_Init_UART1() (same applies to UART2), the UART interface needs to be started to be able to receive or transmit bytes. This can be done by sending the bytes via event to SYS_EVENT_IO_TO_BL↩ UETOOTH (UART1) or by handing over the bytes directly by calling Sys_Writeto_UART1 and Sys_Writeto_UART2. Incoming bytes can be received by defining a reading function with Sys_SetReadingFunction_UART1(pUART_↩ reader) and Sys_SetReadingFunction_UART2(pUART_reader). This function is executed every time a new byte arrives.

6.10.3 License

6.11 Process Manages

Functions to create, switch, block, yield, and terminate processes and start critical sections.

Files

- file process_Management_HDI.c

  Hardware dependent implementations to manage processes (e.g. task swichting)
- file process_Management_HDI.h

  Hardware dependent implementations to manage processes (e.g. task swichting)
- file data.c

  This file includes all functions which are needed to manage data structures needed by the processes management.
- file data.h

  This file includes all functions which are needed to manage data structures needed by the processes management.
- file process_Management.c

  This file includes all functions wich are needed to manage processes (e.g. task swichting)
- file process_Management.h

  This file includes all functions wich are needed to manage processes (e.g. task creation, switching, termination)
- file scheduler.c

  This file includes all functions wich are needed to specify a scheduling algorithm.
- file scheduler.h

  This file includes all functions wich are needed to specify a scheduling algorithm.
- file system_Timer.c

  This file includes all hardware dependent functions, which are nesessary to initialise, configure and run the system Time.
- file system_Timer.h

  This file includes all hardware dependent functions, which are nesessary to initialise, configure and run the system Time.

6.11.1 Detailed Description

Functions to create, switch, block, yield, and terminate processes and start critical sections.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

A process is a basic form to execute functions in OpenSwarm. OpenSwarm does not provide functions to separate memory in pages or segments due to target device architecture. Because all processes are executed in the same memory area, each process can be seen as a single thread and all threads share the same memory. A thread is just represented by a common function. One function can be executed multiple times as individual threads.

OpenSwarm organises processes in three lists of processes (pid sorted):

1. running list: includes all processes are ready to be executed and are scheduled according to the scheduling algorithm.
2. blocked list: includes all processes that are waiting for events to occur.
3. Zombie list: includes all processes that are about to be terminated but not deleted yet.

6.11.2 Usage

The process management is initialised with Sys_Init_Process_Management(void), which generated the System Thread (pid: 0) and initialises all data structures. After initialising, the following functions are available.

6.11.2.1 User code:

1. Processes are started and terminated with Sys_Start_Process(pFunction function) and Sys_Kill_↩ Process(uint16 pid) respectively.
2. A Process can be yield with Sys_Yield(void) and remains in the ready list. The process can be rescheduled by the scheduler.
3. A thread/process can be suspended while waiting for arriving events with Sys_Wait_For_Event(uint16 event↩ ID) and Sys_Wait_For_Condition(uint16 eventID, pConditionFunction function). Processes that are suspended are on the block list and are not rescheduled whilst in it.

6.11.2.2 Internal function (shouldn't be used by the user)

6.11.2.2.1 Scheduling (functions to decide which process is executed at which time)

Functions can be found regarding the scheduling process can be found in scheduler.h and process_Management.h.

- The executing process can be switched by using Sys_Switch_Process(uint16 pid) and Sys_Switch_to_next↩ _Process(void).
- To implement a new scheduling algorithm, struct sys_scheduler_info_s, a function to implement the algorithm (void function(void)), and a function to set the values of the struct (void Sys_Set_Defaults_Info(sys_↩ scheduler_info ∗sct)) needs to be implemented (fund in scheduler.h).

6.11.2.2.2 System Timer (timer to start the scheduling, found in system_Timer.h):

1. The System Timer needs to be initialised and started by Sys_Init_SystemTimer(pFunction) and Sys_Start↩ _SystemTimer(void) respectively (these functions are used when the process Management is initialised and started).
2. It can be stopped, continued, and reset by Sys_Stop_SystemTimer(), Sys_Continue_SystemTimer(), and Sys_Reset_SystemTimer() respectively.
3. The timer can be disabled and enabled (no interrupts) by Sys_Disable_TimerInterrupt(void) and Sys_↩ Enable_TimerInterrupt(void).
4. To force a system timer and therefore an scheduling process, Sys_Force_TimerInterrupt() will cause the system timer interrupt to occur.

6.11.2.2.3 Process Event handling (functions to store/process events with it's subscribed process and add/remove subscriptions) \sa events

- Event subscription to a process can be added and removed by Sys_Add_Event_Subscription and Sys_↩ Remove_Event_Subscription.
- Removing all subscription to any process of a singe event can be done Sys_Remove_All_Event_↩ Subscriptions(uint16 eventID).
- To copy the data of an occurred event to a specific process, Sys_Add_Event_to_Process can be used.
- All stored data is processed by its registered event handler by Sys_Execute_All_EventHandler.
- The event data can be cleared with Sys_Clear_EventData.

6.11.3 Example

```
#include "os/system.h"
#include "os/events/events.h"
#include "os/processes/process_Management.h"

#define WAIT_FOR_ME 0x0F
```

```
Sys_Wait_For_Event(uint16 eventID)

void thread(void){//thread definition
      while(ture){
          //do something as an thread
          sys_event_data * data = Sys_Wait_For_Event(WAIT_FOR_ME);
          Sys_Clear_EventData(data);
      }
}

int main(void){
 //initialise some global or local variables

 int variable;

    Sys_Init_Kernel();

 Sys_Register_Event(WAIT_FOR_ME);

 Sys_Start_Kernel();
    while(1){

     if( condition ){
          Sys_Send_Event(WAIT_FOR_ME, &variable, sizeof(int));
     }
     //do something
    }
}
```

6.11.4 License

Chapter 7

Data Structure Documentation

7.1 sys_event_data_s Struct Reference

This struct contains data of the size **size** at the memory of **value**. It is a struct for a linked list.

```
#include <events.h>
```

Collaboration diagram for sys_event_data_s:

sys_event_data_s
next

Data Fields

- void ∗ value
- uint16 size
- struct sys_event_data_s ∗ next

7.1.1 Detailed Description

This struct contains data of the size **size** at the memory of **value**. It is a struct for a linked list.

Definition at line 89 of file events.h.

7.1.2 Field Documentation

7.1.2.1 struct sys_event_data_s∗ sys_event_data_s::next

pointer to the next element in the List

Definition at line 93 of file events.h.

7.1.2.2 uint16 sys_event_data_s::size

size of the dransfered data (bytes)

Definition at line 91 of file events.h.

7.1.2.3 void∗ sys_event_data_s::value

pointer to the data transfered by an event

Definition at line 90 of file events.h.

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

- events/events.h

7.2 sys_i2c_message_s Struct Reference

Linked list element of messages that need to be sent via I2C.

`#include <i2c_data.h>`

Collaboration diagram for sys_i2c_message_s:

sys_i2c_message_s
next

Data Fields

- uint8 i2c_device_address
- uint8 ∗ data
- uint16 length
- bool write
- pByteFunction handler
- struct sys_i2c_message_s ∗ next

7.2.1 Detailed Description

Linked list element of messages that need to be sent via I2C.

Definition at line 32 of file i2c_data.h.

7.2.2 Field Documentation

7.2.2.1 uint8∗ sys_i2c_message_s::data

Definition at line 34 of file i2c_data.h.

7.2.2.2 pByteFunction sys_i2c_message_s::handler

Definition at line 37 of file i2c_data.h.

7.2.2.3 uint8 sys_i2c_message_s::i2c_device_address

Definition at line 33 of file i2c_data.h.

7.2.2.4 uint16 sys_i2c_message_s::length

Definition at line 35 of file i2c_data.h.

7.2.2.5 struct sys_i2c_message_s∗ sys_i2c_message_s::next

Definition at line 38 of file i2c_data.h.

7.2.2.6 bool sys_i2c_message_s::write

Definition at line 36 of file i2c_data.h.

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

- platform/e-puck/i2c_data.h

7.3 sys_motors_s Struct Reference

This struct contains speed for motors.

Data Fields

- sint16 speed

7.3.1 Detailed Description

This struct contains speed for motors.

Definition at line 33 of file motors.c.

7.3.2 Field Documentation

7.3.2.1 sint16 sys_motors_s::speed

Definition at line 34 of file motors.c.

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

- platform/e-puck/motors.c

7.4 sys_occured_event_s Struct Reference

List of occured events.

```
#include <data.h>
```

Collaboration diagram for sys_occured_event_s:

Data Fields

- uint16 eventID
- struct sys_occured_event_s ∗ next

7.4.1 Detailed Description

List of occured events.

This struct sores the event ID of an occurred event

Definition at line 34 of file data.h.

7.4.2 Field Documentation

7.4.2.1 uint16 sys_occured_event_s::eventID

Definition at line 35 of file data.h.

7.4.2.2 struct sys_occured_event_s∗ sys_occured_event_s::next

Definition at line 37 of file data.h.

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

- processes/data.h

7.5 sys_periodical_IOHandler_s Struct Reference

```
#include <io_HDI.h>
```

Collaboration diagram for sys_periodical_IOHandler_s:

sys_periodical_IOHandler_s
next

Data Fields

- pFunction function
- struct sys_periodical_IOHandler_s ∗ next

7.5.1 Detailed Description

Definition at line 28 of file io_HDI.h.

7.5.2 Field Documentation

7.5.2.1 pFunction sys_periodical_IOHandler_s::function

Definition at line 29 of file io_HDI.h.

7.5.2.2 struct sys_periodical_IOHandler_s∗ sys_periodical_IOHandler_s::next

Definition at line 31 of file io_HDI.h.

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

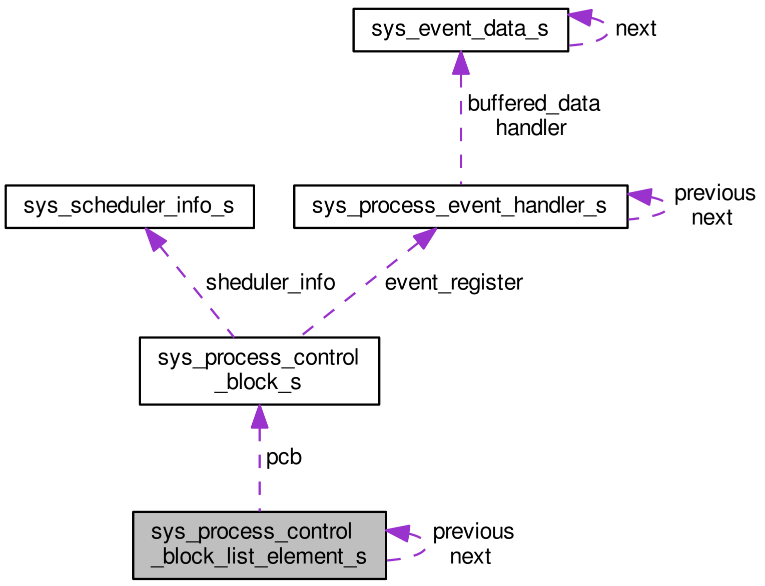
- platform/e-puck/io_HDI.h

7.6 sys_process_control_block_list_element_s Struct Reference

Container struct for Process Control Block.

```
#include <data.h>
```

Collaboration diagram for sys_process_control_block_list_element_s:



Data Fields

- sys_process_control_block pcb
- struct sys_process_control_block_list_element_s ∗ previous
- struct sys_process_control_block_list_element_s ∗ next

7.6.1 Detailed Description

Container struct for Process Control Block.

This struct is a container (linked list) for PCB

Definition at line 77 of file data.h.

7.6.2 Field Documentation

7.6.2.1 struct **sys_process_control_block_list_element_s**∗ sys_process_control_block_list_element_s::next

Definition at line 82 of file data.h.

7.6.2.2 **sys_process_control_block** sys_process_control_block_list_element_s::pcb

Definition at line 79 of file data.h.

7.6.2.3 struct **sys_process_control_block_list_element_s**∗ sys_process_control_block_list_element_s::previous

Definition at line 81 of file data.h.

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

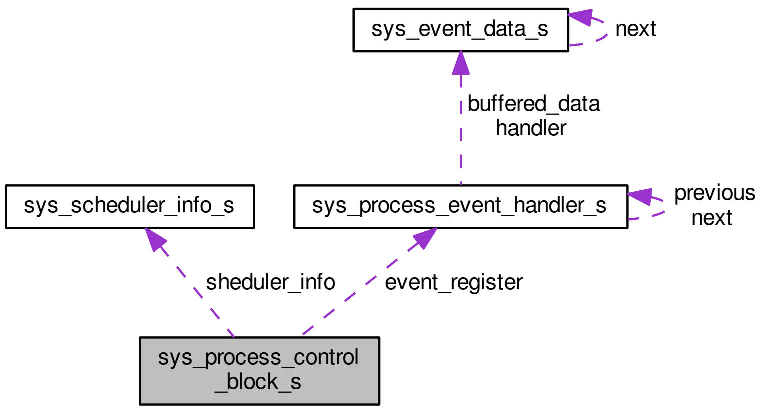
- processes/data.h

7.7 sys_process_control_block_s Struct Reference

Process Control Block for the processes.

`#include <data.h>`

Collaboration diagram for sys_process_control_block_s:



Data Fields

- uint16 process_ID
- uint16 stackPointer
- uint16 framePointer
- uint16 stackPointerLimit
- sys_scheduler_info sheduler_info
- sys_process_event_handler ∗ event_register
- uint16 ∗ process_stack

7.7.1 Detailed Description

Process Control Block for the processes.

This struct sores all information of the current state of a process

Definition at line 58 of file data.h.

7.7.2 Field Documentation

7.7.2.1 sys_process_event_handler∗ sys_process_control_block_s::event_register

Definition at line 66 of file data.h.

7.7.2.2 uint16 sys_process_control_block_s::framePointer

Definition at line 62 of file data.h.

7.7.2.3 uint16 sys_process_control_block_s::process_ID

Definition at line 60 of file data.h.

7.7.2.4 uint16∗ sys_process_control_block_s::process_stack

Definition at line 68 of file data.h.

7.7.2.5 sys_scheduler_info sys_process_control_block_s::sheduler_info

Definition at line 65 of file data.h.

7.7.2.6 uint16 sys_process_control_block_s::stackPointer

Stack Pointer to TOP

Definition at line 61 of file data.h.

7.7.2.7 uint16 sys_process_control_block_s::stackPointerLimit

Stack Pointer + MAX SIZE

Definition at line 63 of file data.h.

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

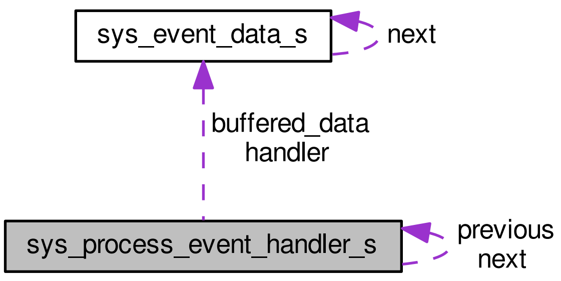
- processes/data.h

7.8 sys_process_event_handler_s Struct Reference

List of process event-handlers.

```
#include <data.h>
```

Collaboration diagram for sys_process_event_handler_s:



Data Fields

- uint16 eventID
- pEventHandlerFunction handler
- pConditionFunction condition
- sys_event_data ∗ buffered_data
- struct sys_process_event_handler_s ∗ previous
- struct sys_process_event_handler_s ∗ next

7.8.1 Detailed Description

List of process event-handlers.

This struct sores all information needed to decide if the event-handler is executed for the event (eventID). To store the event data and be executed, a condition has to be met.

Definition at line 44 of file data.h.

7.8.2 Field Documentation

7.8.2.1 sys_event_data∗ sys_process_event_handler_s::buffered_data

stores the data

Definition at line 48 of file data.h.

7.8.2.2 pConditionFunction sys_process_event_handler_s::condition

Pointer to function which checks if the event-handler has to be executed (true) or nor (false)

Definition at line 47 of file data.h.

7.8.2.3 uint16 sys_process_event_handler_s::eventID

Definition at line 45 of file data.h.

7.8.2.4 pEventHandlerFunction sys_process_event_handler_s::handler

Pointer to function which computes the evnt data

Definition at line 46 of file data.h.

7.8.2.5 struct sys_process_event_handler_s∗ sys_process_event_handler_s::next

Definition at line 51 of file data.h.

7.8.2.6 struct sys_process_event_handler_s∗ sys_process_event_handler_s::previous

Definition at line 50 of file data.h.

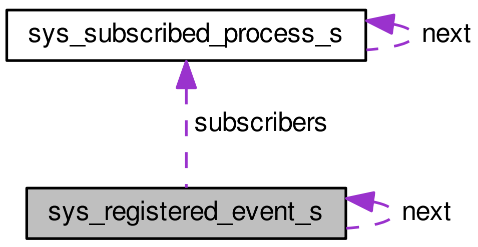
The documentation for this struct was generated from the following file:

- processes/data.h

7.9 sys_registered_event_s Struct Reference

A struct to store registered events. It also includes a list of processes that are subscribed to the registered event.

Collaboration diagram for sys_registered_event_s:



Data Fields

- uint16 id
- sys_subscribed_process ∗ subscribers
- struct sys_registered_event_s ∗ next

7.9.1 Detailed Description

A struct to store registered events. It also includes a list of processes that are subscribed to the registered event.

Definition at line 32 of file events.c.

7.9.2 Field Documentation

7.9.2.1 uint16 sys_registered_event_s::id

event identifier

Definition at line 33 of file events.c.

7.9.2.2 struct sys_registered_event_s∗ sys_registered_event_s::next

pointer to the next element in the List

Definition at line 35 of file events.c.

7.9.2.3 sys_subscribed_process∗ sys_registered_event_s::subscribers

pointer to a list of subscribed processes

Definition at line 34 of file events.c.

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

- events/events.c

7.10 sys_rgb_pixel_s Struct Reference

This bitfield contains the structure of the received pixel of a camera.

```
#include <camera.h>
```

Data Fields

- uint8 red: 5
- uint8 green: 6
- uint8 blue: 5

7.10.1 Detailed Description

This bitfield contains the structure of the received pixel of a camera.

Definition at line 57 of file camera.h.

7.10.2 Field Documentation

7.10.2.1 uint8 sys_rgb_pixel_s::blue

Definition at line 60 of file camera.h.

7.10.2.2 uint8 sys_rgb_pixel_s::green

Definition at line 59 of file camera.h.

7.10.2.3 uint8 sys_rgb_pixel_s::red

Definition at line 58 of file camera.h.

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

- platform/e-puck/camera.h

7.11 sys_scheduler_info_s Struct Reference

The scheduling information for each process.

```
#include <scheduler.h>
```

Data Fields

- unsigned short state
- unsigned short priority

7.11.1 Detailed Description

The scheduling information for each process.

This struct defines all values wich are needed for the scheduling algorithm

Definition at line 37 of file scheduler.h.

7.11.2 Field Documentation

7.11.2.1 unsigned short sys_scheduler_info_s::priority

process priority level

Definition at line 39 of file scheduler.h.

7.11.2.2 unsigned short sys_scheduler_info_s::state

Process state information

Definition at line 38 of file scheduler.h.

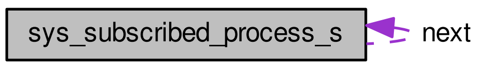
The documentation for this struct was generated from the following file:

- processes/scheduler.h

7.12 sys_subscribed_process_s Struct Reference

A struct to store a list of subscribed processes.

Collaboration diagram for sys_subscribed_process_s:



Data Fields

- uint16 pid
- struct sys_subscribed_process_s ∗ next

7.12.1 Detailed Description

A struct to store a list of subscribed processes.

Definition at line 24 of file events.c.

7.12.2 Field Documentation

7.12.2.1 struct sys_subscribed_process_s∗ sys_subscribed_process_s::next

pointer to the next element in the List

Definition at line 26 of file events.c.

7.12.2.2 uint16 sys_subscribed_process_s::pid

process identifier

Definition at line 25 of file events.c.

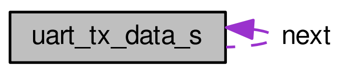
The documentation for this struct was generated from the following file:

- events/events.c

7.13 uart_tx_data_s Struct Reference

```
#include <uart_HDI.h>
```

Collaboration diagram for uart_tx_data_s:



Data Fields

- uint8 ∗ data
- uint16 length
- struct uart_tx_data_s ∗ next

7.13.1 Detailed Description

Linked list element to store the transmission data

This struct contains data and the amount of bytes that should be sent via UART1.

Definition at line 47 of file uart_HDI.h.

7.13.2 Field Documentation

7.13.2.1 uint8∗ uart_tx_data_s::data

Definition at line 48 of file uart_HDI.h.

7.13.2.2 uint16 uart_tx_data_s::length

Definition at line 49 of file uart_HDI.h.

7.13.2.3 struct uart_tx_data_s∗ uart_tx_data_s::next

Definition at line 51 of file uart_HDI.h.

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

- platform/e-puck/uart_HDI.h

Chapter 8

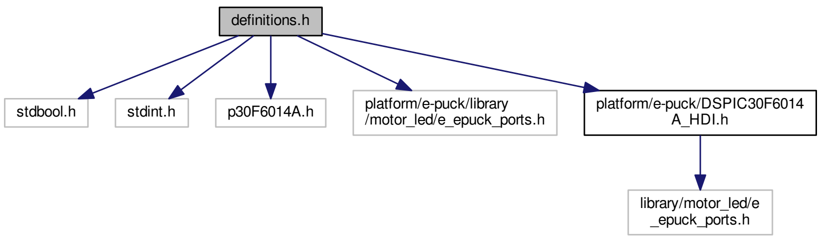
File Documentation

8.1 definitions.h File Reference

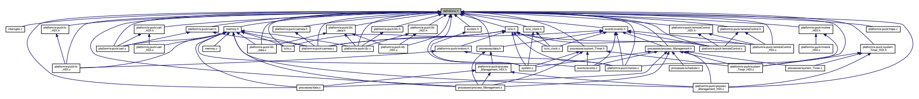
This file defines all preprocessor variables and project wide types.

```
#include <stdbool.h>
#include <stdint.h>
#include <p30F6014A.h>
#include "platform/e-puck/library/motor_led/e_epuck_ports.h"
#include "platform/e-puck/DSPIC30F6014A_HDI.h"
```

Include dependency graph for definitions.h:


definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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



Macros

- #define EPUCK_USED
- #define UART1_RX _RF2
- #define UART1_TX _RF3
- #define UART2_RX _RF4
- #define UART2_TX _RF5
- #define UART1_RX_DIR _TRISF2
- #define UART1_TX_DIR _TRISF3

- #define UART2_RX_DIR _TRISF4
- #define UART2_TX_DIR _TRISF5
- #define RC_BUTTON_STANDBY 12
- #define RC_BUTTON_SCREEN 11
- #define RC_BUTTON_LANG 15
- #define RC_BUTTON_SUBTTL 31
- #define RC_BUTTON_INTERNET 46
- #define RC_BUTTON_RED 55
- #define RC_BUTTON_GREEN 54
- #define RC_BUTTON_YELLOW 50
- #define RC_BUTTON_BLUE 52
- #define RC_BUTTON_0 0
- #define RC_BUTTON_1 1
- #define RC_BUTTON_2 2
- #define RC_BUTTON_3 3
- #define RC_BUTTON_4 4
- #define RC_BUTTON_5 5
- #define RC_BUTTON_6 6
- #define RC_BUTTON_7 7
- #define RC_BUTTON_8 8
- #define RC_BUTTON_9 9
- #define RC_BUTTON_TELE_TEXT 60
- #define RC_BUTTON_SWAP 34
- #define RC_BUTTON_OK 53
- #define RC_BUTTON_CURSOR_UP 20
- #define RC_BUTTON_CURSOR_DOWN 19
- #define RC_BUTTON_CURSOR_LEFT 21
- #define RC_BUTTON_CURSOR_RIGHT 22
- #define RC_BUTTON_BACK 10
- #define RC_BUTTON_MENU 48
- #define RC_BUTTON_EPG 47
- #define RC_BUTTON_FAV 40
- #define RC_BUTTON_SOURCE 56
- #define RC_BUTTON_INFO 18
- #define RC_BUTTON_PRESETS 14
- #define RC_BUTTON_SLEEP 42
- #define RC_BUTTON_VOLUME_UP 16
- #define RC_BUTTON_VOLUME_DOWN 17
- #define RC_BUTTON_MUTE 13
- #define RC_BUTTON_CHANNEL_UP 32
- #define RC_BUTTON_CHANNEL_DOWN 33
- #define RC_BUTTON_PAUSE 48
- #define RC_BUTTON_REWIND 50
- #define RC_BUTTON_WIND 52
- #define RC_BUTTON_PLAY 53
- #define RC_BUTTON_STOP 54
- #define RC_BUTTON_RECORD 55
- #define SYS_EVENT_TERMINATION 0x01
- #define SYS_EVENT_IO_MOTOR_LEFT 0x02
- #define SYS_EVENT_IO_MOTOR_RIGHT 0x03
- #define SYS_EVENT_IO_CAMERA 0x04
- #define SYS_EVENT_IO_REMOECONTROL 0x05
- #define SYS_EVENT_IO_TO_BLUETOOTH 0x06
- #define SYS_EVENT_1ms_CLOCK 0x07
- #define ALL_FUNCTIONS ((pEventHandlerFunction) 0xFFFFFFFE)

Typedefs

- typedef enum sys_colour sys_colour

  defines a system-wide colour definition
- typedef unsigned char uint8
- typedef unsigned short uint16
- typedef unsigned int uint32
- typedef signed char sint8
- typedef signed short sint16
- typedef signed int sint32
- typedef signed short sint
- typedef unsigned short uint
- typedef void(∗ pFunction) (void)
- typedef void(∗ pByteFunction) (uint8)
- typedef void(∗ pUART_reader) (uint8 data)

Enumerations

- enum sys_colour {
  BLACK = 0b00000000, RED = 0b00000100, YELLOW = 0b00000110, GREEN = 0b00000010,
  CYAN = 0b00000011, BLUE = 0b00000001, MAGENTA = 0b00000101, WHITE = 0b00000111 }

  defines a system-wide colour definition

8.1.1 Detailed Description

This file defines all preprocessor variables and project wide types.

Author

Stefan M. Trenkwalder

Version

1.0

Date

2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.1.2 Macro Definition Documentation

8.1.2.1 #define ALL_FUNCTIONS ((pEventHandlerFunction) 0xFFFFFFFE)

Definition at line 111 of file definitions.h.

8.1.2.2 #define EPUCK_USED

Definition at line 19 of file definitions.h.

8.1.2.3 #define RC_BUTTON_0 0

Definition at line 61 of file definitions.h.

8.1.2.4 #define RC_BUTTON_1 1

Definition at line 62 of file definitions.h.

8.1.2.5 #define RC_BUTTON_2 2

Definition at line 63 of file definitions.h.

8.1.2.6 #define RC_BUTTON_3 3

Definition at line 64 of file definitions.h.

8.1.2.7 #define RC_BUTTON_4 4

Definition at line 65 of file definitions.h.

8.1.2.8 #define RC_BUTTON_5 5

Definition at line 66 of file definitions.h.

8.1.2.9 #define RC_BUTTON_6 6

Definition at line 67 of file definitions.h.

8.1.2.10 #define RC_BUTTON_7 7

Definition at line 68 of file definitions.h.

8.1.2.11 #define RC_BUTTON_8 8

Definition at line 69 of file definitions.h.

8.1.2.12 #define RC_BUTTON_9 9

Definition at line 70 of file definitions.h.

8.1.2.13 #define RC_BUTTON_BACK 10

Definition at line 79 of file definitions.h.

8.1.2.14 #define RC_BUTTON_BLUE 52

Definition at line 59 of file definitions.h.

8.1.2.15 #define RC_BUTTON_CHANNEL_DOWN 33

Definition at line 93 of file definitions.h.

8.1.2.16 #define RC_BUTTON_CHANNEL_UP 32

Definition at line 92 of file definitions.h.

8.1.2.17 #define RC_BUTTON_CURSOR_DOWN 19

Definition at line 76 of file definitions.h.

8.1.2.18 #define RC_BUTTON_CURSOR_LEFT 21

Definition at line 77 of file definitions.h.

8.1.2.19 #define RC_BUTTON_CURSOR_RIGHT 22

Definition at line 78 of file definitions.h.

8.1.2.20 #define RC_BUTTON_CURSOR_UP 20

Definition at line 75 of file definitions.h.

8.1.2.21 #define RC_BUTTON_EPG 47

Definition at line 81 of file definitions.h.

8.1.2.22 #define RC_BUTTON_FAV 40

Definition at line 82 of file definitions.h.

8.1.2.23 #define RC_BUTTON_GREEN 54

Definition at line 57 of file definitions.h.

8.1.2.24 #define RC_BUTTON_INFO 18

Definition at line 85 of file definitions.h.

8.1.2.25 #define RC_BUTTON_INTERNET 46

Definition at line 54 of file definitions.h.

8.1.2.26 #define RC_BUTTON_LANG 15

Definition at line 52 of file definitions.h.

8.1.2.27 #define RC_BUTTON_MENU 48

Definition at line 80 of file definitions.h.

8.1.2.28 #define RC_BUTTON_MUTE 13

Definition at line 91 of file definitions.h.

8.1.2.29 #define RC_BUTTON_OK 53

Definition at line 74 of file definitions.h.

8.1.2.30 #define RC_BUTTON_PAUSE 48

Definition at line 96 of file definitions.h.

8.1.2.31 #define RC_BUTTON_PLAY 53

Definition at line 99 of file definitions.h.

8.1.2.32 #define RC_BUTTON_PRESETS 14

Definition at line 86 of file definitions.h.

8.1.2.33 #define RC_BUTTON_RECORD 55

Definition at line 101 of file definitions.h.

8.1.2.34 #define RC_BUTTON_RED 55

Definition at line 56 of file definitions.h.

8.1.2.35 #define RC_BUTTON_REWIND 50

Definition at line 97 of file definitions.h.

8.1.2.36 #define RC_BUTTON_SCREEN 11

Definition at line 51 of file definitions.h.

8.1.2.37 #define RC_BUTTON_SLEEP 42

Definition at line 87 of file definitions.h.

8.1.2.38 #define RC_BUTTON_SOURCE 56

Definition at line 84 of file definitions.h.

8.1.2.39 #define RC_BUTTON_STANDBY 12

Definition at line 49 of file definitions.h.

8.1.2.40 #define RC_BUTTON_STOP 54

Definition at line 100 of file definitions.h.

8.1.2.41 #define RC_BUTTON_SUBTTL 31

Definition at line 53 of file definitions.h.

8.1.2.42 #define RC_BUTTON_SWAP 34

Definition at line 72 of file definitions.h.

8.1.2.43 #define RC_BUTTON_TELE_TEXT 60

Definition at line 71 of file definitions.h.

8.1.2.44 #define RC_BUTTON_VOLUME_DOWN 17

Definition at line 90 of file definitions.h.

8.1.2.45 #define RC_BUTTON_VOLUME_UP 16

Definition at line 89 of file definitions.h.

8.1.2.46 #define RC_BUTTON_WIND 52

Definition at line 98 of file definitions.h.

8.1.2.47 #define RC_BUTTON_YELLOW 50

Definition at line 58 of file definitions.h.

8.1.2.48 #define SYS_EVENT_1ms_CLOCK 0x07

Definition at line 109 of file definitions.h.

8.1.2.49 #define SYS_EVENT_IO_CAMERA 0x04

Definition at line 106 of file definitions.h.

8.1.2.50 #define SYS_EVENT_IO_MOTOR_LEFT 0x02

Definition at line 104 of file definitions.h.

8.1.2.51 #define SYS_EVENT_IO_MOTOR_RIGHT 0x03

Definition at line 105 of file definitions.h.

8.1.2.52 #define SYS_EVENT_IO_REMOECONTROL 0x05

Definition at line 107 of file definitions.h.

8.1.2.53 #define SYS_EVENT_IO_TO_BLUETOOTH 0x06

Definition at line 108 of file definitions.h.

8.1.2.54 #define SYS_EVENT_TERMINATION 0x01

Definition at line 103 of file definitions.h.

8.1.2.55 #define UART1_RX _RF2

Definition at line 37 of file definitions.h.

8.1.2.56 #define UART1_RX_DIR _TRISF2

Definition at line 42 of file definitions.h.

8.1.2.57 #define UART1_TX _RF3

Definition at line 38 of file definitions.h.

8.1.2.58 #define UART1_TX_DIR _TRISF3

Definition at line 43 of file definitions.h.

8.1.2.59 #define UART2_RX _RF4

Definition at line 39 of file definitions.h.

8.1.2.60 #define UART2_RX_DIR _TRISF4

Definition at line 44 of file definitions.h.

8.1.2.61 #define UART2_TX _RF5

Definition at line 40 of file definitions.h.

8.1.2.62 #define UART2_TX_DIR _TRISF5

Definition at line 45 of file definitions.h.

8.1.3 Typedef Documentation

8.1.3.1 typedef void(∗ pByteFunction) (uint8)

Defines a pointer to a function with no return value and one argument

Definition at line 141 of file definitions.h.

8.1.3.2 typedef void(∗ pFunction) (void)

Defines a pointer to a function with no return value and argument

Definition at line 140 of file definitions.h.

8.1.3.3 typedef void(∗ pUART_reader) (uint8 data)

Defines a pointer to a function with no return value and one argument

Definition at line 143 of file definitions.h.

8.1.3.4 typedef signed short sint

Definition at line 136 of file definitions.h.

8.1.3.5 typedef signed short sint16

Defines a signed 16bit integer

Definition at line 132 of file definitions.h.

8.1.3.6 typedef signed int sint32

Defines a signed 32bit integer

Definition at line 133 of file definitions.h.

8.1.3.7 typedef signed char sint8

Defines a signed 8bit integer

Definition at line 131 of file definitions.h.

8.1.3.8 typedef enum sys_colour sys_colour

defines a system-wide colour definition

This enum defines a system-wide colour. (it is based on one bit for red, blue, and green). In total, 8 colours are defined with the first three bits.

8.1.3.9 typedef unsigned short uint

Definition at line 137 of file definitions.h.

8.1.3.10 typedef unsigned short uint16

Defines an unsigned 16bit integer

Definition at line 129 of file definitions.h.

8.1.3.11 typedef unsigned int uint32

Defines an unsigned 32bit integer

Definition at line 130 of file definitions.h.

8.1.3.12 typedef unsigned char uint8

Defines an unsigned 8bit integer

Definition at line 128 of file definitions.h.

8.1.4 Enumeration Type Documentation

8.1.4.1 enum sys_colour

defines a system-wide colour definition

This enum defines a system-wide colour. (it is based on one bit for red, blue, and green). In total, 8 colours are defined with the first three bits.

Enumerator

BLACK

RED

YELLOW

GREEN

CYAN

BLUE

MAGENTA

WHITE

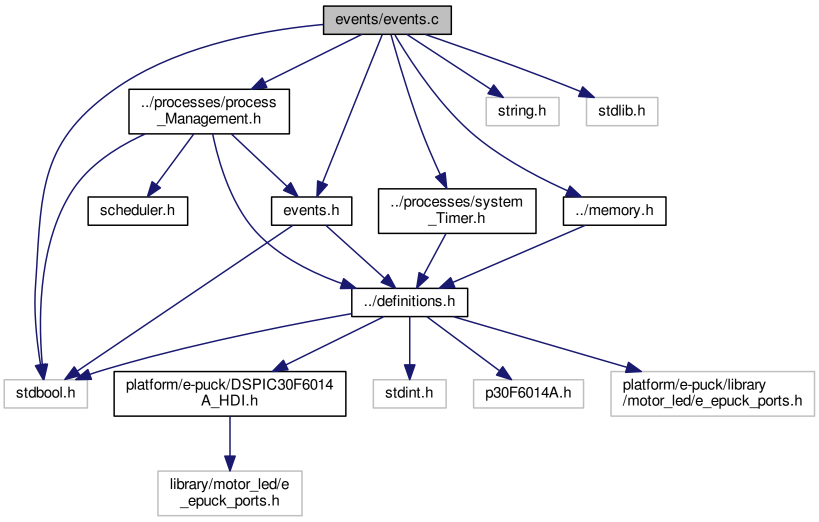
Definition at line 118 of file definitions.h.

8.2 events/events.c File Reference

This file includes all system calls needed to create, (un)subscribe, (un)register, and delete events and related handler.

```
#include "events.h"
#include "../processes/process_Management.h"
#include "../processes/system_Timer.h"
#include "../memory.h"
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
```

Include dependency graph for events.c:


events/events.c
../processes/process _Management.h
string.h
stdlib.h
scheduler.h
events.h
../processes/system _Timer.h
../memory.h
../definitions.h
stdbool.h
platform/e-puck/DSPIC30F6014 A_HDI.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
library/motor_led/e _epuck_ports.h

Data Structures

- struct sys_subscribed_process_s

  A struct to store a list of subscribed processes.
- struct sys_registered_event_s

  A struct to store registered events. It also includes a list of processes that are subscribed to the registered event.

Typedefs

- typedef struct sys_subscribed_process_s sys_subscribed_process

  A struct to store a list of subscribed processes.
- typedef struct sys_registered_event_s sys_registered_event

  A struct to store registered events. It also includes a list of processes that are subscribed to the registered event.

Functions

- sys_registered_event ∗ Sys_Find_Event (uint16 eventID)
- bool Sys_Send_Event (uint16 eventID, void ∗data, uint16 data_size)
- bool Sys_Send_IntEvent (uint16 eventID, uint16 data)
- bool Sys_Register_Event (uint16 eventID)
- bool Sys_Subscribe_to_Event (uint16 eventID, uint16 pid, pEventHandlerFunction handler, pCondition←Function condition)
- void Sys_Unregister_Event (uint16 eventID)
- void Sys_Unsubscribe_from_Event (uint16 eventID, uint16 pid)
- void Sys_Unsubscribe_Handler_from_Event (uint16 eventID, pEventHandlerFunction func, uint16 pid)
- bool Sys_IsEventRegistered (uint16 eventID)
- void Sys_Unsubscribe_Process (uint16 pid)

Variables

- sys_registered_event ∗ registered_events = 0

8.2.1 Detailed Description

This file includes all system calls needed to create, (un)subscribe, (un)register, and delete events and related handler.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

23 March 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.2.2 Typedef Documentation

8.2.2.1 typedef struct **sys_registered_event_s sys_registered_event**

A struct to store registered events. It also includes a list of processes that are subscribed to the registered event.

8.2.2.2 typedef struct **sys_subscribed_process_s sys_subscribed_process**

A struct to store a list of subscribed processes.

8.2.3 Function Documentation

8.2.3.1 **sys_registered_event** ∗ Sys_Find_Event (**uint16** *eventID*)

finds the registered event

This function returns the data structure of an event if the eventID was registered otherwise it's 0.

Parameters

`in`	*eventID*	ID of the event

Returns

pointer to the data structure of the found event (or 0 if it wasn't found)

Definition at line 313 of file events.c.

8.2.3.2 bool Sys_IsEventRegistered (**uint16** *eventID*)

returns true if the event was registered

returns true if the event was registered

Parameters

in	*eventID*	ID of the event

Returns

is the event registered?

Definition at line 335 of file events.c.

8.2.3.3 bool Sys_Register_Event (uint16 *eventID*)

Function to register an event

This function registers an new event. The registration tells the operating system that this event can occur.

Parameters

in	*eventID*	ID of the event

Returns

was it successful.

Definition at line 101 of file events.c.

8.2.3.4 bool Sys_Send_Event (uint16 *eventID,* void ∗ *data,* uint16 *data_size*)

Function to send an event

This function sends an event to all subscribers.

Parameters

in	*eventID*	ID of the event
in	*data*	pointer to the data that want to be sent as an event
in	*data_size*	size of the data in bytes

Returns

was it successful.

Definition at line 60 of file events.c.

8.2.3.5 bool Sys_Send_IntEvent (uint16 *eventID,* uint16 *data*) [inline]

Function to send an integer event

This function sends an integer (16-bit) to all subscribers.

Parameters

in	*eventID*	ID of the event
in	*data*	integer value that should be sent as an event

Returns

was it successful.

Definition at line 88 of file events.c.

8.2.3.6 bool Sys_Subscribe_to_Event (uint16 *eventID,* uint16 *pid,* pEventHandlerFunction *handler,* pConditionFunction *condition*)

subscribes a specific handler function to an process and a specific event

This function subscribes a specific handler function to an process and a specific event

Parameters

`in`	*eventID*	ID of the event
`in`	*pid*	ID of the process
`in`	*handler*	pointer to the function that should handle the event data
`in`	*condition*	pointer to the function that decides if the handler should be executed or not

Returns

was it successful.

Definition at line 144 of file events.c.

8.2.3.7 void Sys_Unregister_Event (uint16 *eventID*)

unregisters an event

This function unregisters an event

Parameters

`in`	*eventID*	ID of the event

Definition at line 190 of file events.c.

8.2.3.8 void Sys_Unsubscribe_from_Event (uint16 *eventID,* uint16 *pid*)

unsubscribes an event

This function unsubscribes an event

Parameters

`in`	*eventID*	ID of the event
`in`	*pid*	ID of the process

Definition at line 242 of file events.c.

8.2.3.9 void Sys_Unsubscribe_Handler_from_Event (uint16 *eventID,* pEventHandlerFunction *func,* uint16 *pid*)

only unsubscribes a specific handler function

This function only unsubscribes a specific handler function

Parameters

`in`	*eventID*	ID of the event
`in`	*func*	pointer to the handler function
`in`	*pid*	ID of the process

Definition at line 278 of file events.c.

8.2.3.10 void Sys_Unsubscribe_Process (uint16 *pid*)

unsubscribes all events that were subscribed to a process

unsubscribes all events that were subscribed to a process

Parameters

`in`	*pid*	process identifier

Definition at line 356 of file events.c.

8.2.4 Variable Documentation

8.2.4.1 sys_registered_event∗ registered_events = 0

pointer to the List of registered events

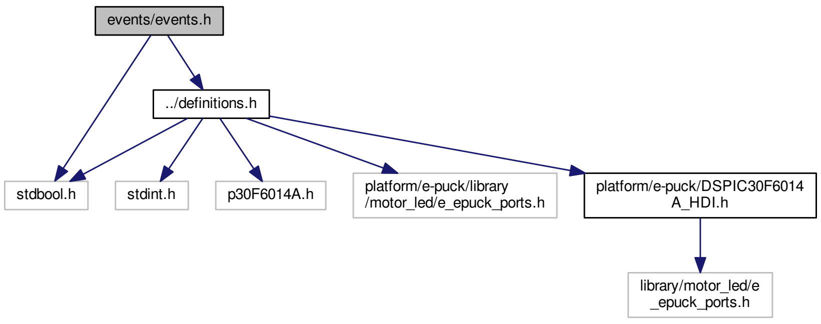
Definition at line 46 of file events.c.

8.3 events/events.h File Reference

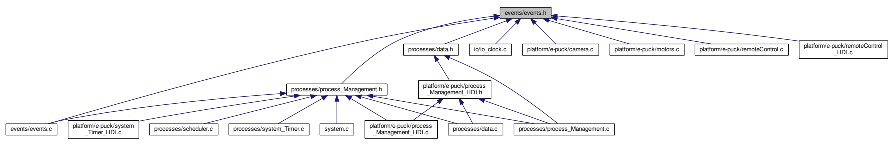
functions to create handle and configure events.

```
#include <stdbool.h>
#include "../definitions.h"
```

Include dependency graph for events.h:



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



Data Structures

- struct sys_event_data_s

    *This struct contains data of the size **size** at the memory of **value**. It is a struct for a linked list.*

Typedefs

- typedef struct sys_event_data_s sys_event_data

*This struct contains data of the size **size** at the memory of **value**. It is a struct for a linked list.*

- typedef bool(∗ pEventHandlerFunction) (uint16, uint16, sys_event_data ∗)

  Event handler function pinter type (process id, event id, received data)
- typedef bool(∗ pConditionFunction) (void ∗)

  Condition function pinter type (received data)

Functions

- bool Sys_Send_Event (uint16 eventID, void ∗data, uint16 data_size)
- bool Sys_Send_IntEvent (uint16 eventID, uint16 data)
- bool Sys_Register_Event (uint16 eventID)
- void Sys_Unregister_Event (uint16 eventID)
- bool Sys_Subscribe_to_Event (uint16 eventID, uint16 pid, pEventHandlerFunction handler, pCondition←Function condition)
- void Sys_Unsubscribe_from_Event (uint16 eventID, uint16 pid)
- void Sys_Unsubscribe_Process (uint16 pid)
- bool Sys_IsEventRegistered (uint16 eventID)

8.3.1 Detailed Description

functions to create handle and configure events.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

23 March 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.3.2 Typedef Documentation

8.3.2.1 typedef bool(∗ pConditionFunction) (void ∗)

Condition function pinter type (received data)

Definition at line 105 of file events.h.

8.3.2.2 typedef bool(∗ pEventHandlerFunction) (uint16, uint16, sys_event_data ∗)

Event handler function pinter type (process id, event id, received data)

Definition at line 100 of file events.h.

8.3.2.3 typedef struct sys_event_data_s sys_event_data

This struct contains data of the size **size** at the memory of **value**. It is a struct for a linked list.

8.3.3 Function Documentation

8.3.3.1 bool Sys_IsEventRegistered (uint16 *eventID*)

returns true if the event was registered

returns true if the event was registered

Parameters

`in`	*eventID*	ID of the event

Returns

is the event registered?

Definition at line 335 of file events.c.

8.3.3.2 bool Sys_Register_Event (uint16 *eventID*)

Function to register an event

This function registers an new event. The registration tells the operating system that this event can occur.

Parameters

`in`	*eventID*	ID of the event

Returns

was it successful.

Definition at line 101 of file events.c.

8.3.3.3 bool Sys_Send_Event (uint16 *eventID,* void $*$ *data,* uint16 *data_size*)

Function to send an event

This function sends an event to all subscribers.

Parameters

`in`	*eventID*	ID of the event
`in`	*data*	pointer to the data that want to be sent as an event
`in`	*data_size*	size of the data in bytes

Returns

was it successful.

Definition at line 60 of file events.c.

8.3.3.4 bool Sys_Send_IntEvent (uint16 *eventID,* uint16 *data*) `[inline]`

Function to send an integer event

This function sends an integer (16-bit) to all subscribers.

Parameters

`in`	*eventID*	ID of the event
`in`	*data*	integer value that should be sent as an event

Returns

was it successful.

Definition at line 88 of file events.c.

8.3.3.5 bool Sys_Subscribe_to_Event (uint16 *eventID*, uint16 *pid*, pEventHandlerFunction *handler*, pConditionFunction *condition*)

subscribes a specific handler function to an process and a specific event

This function subscribes a specific handler function to an process and a specific event

Parameters

`in`	*eventID*	ID of the event
`in`	*pid*	ID of the process
`in`	*handler*	pointer to the function that should handle the event data
`in`	*condition*	pointer to the function that decides if the handler should be executed or not

Returns

was it successful.

Definition at line 144 of file events.c.

8.3.3.6 void Sys_Unregister_Event (uint16 *eventID*)

unregisters an event

This function unregisters an event

Parameters

`in`	*eventID*	ID of the event

Definition at line 190 of file events.c.

8.3.3.7 void Sys_Unsubscribe_from_Event (uint16 *eventID*, uint16 *pid*)

unsubscribes an event

This function unsubscribes an event

Parameters

`in`	*eventID*	ID of the event
`in`	*pid*	ID of the process

Definition at line 242 of file events.c.

8.3.3.8 void Sys_Unsubscribe_Process (uint16 *pid*)

unsubscribes all events that were subscribed to a process

unsubscribes all events that were subscribed to a process

Parameters

in	*pid*	process identifier

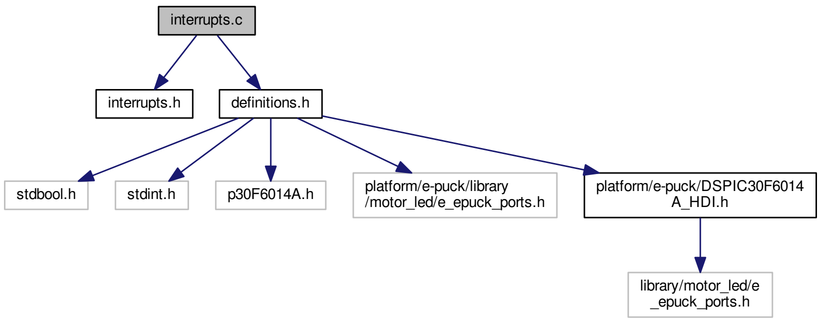
Definition at line 356 of file events.c.

8.4 interrupts.c File Reference

includes basic system calls to create atomic sections. (Sections that cannot be interrupted)

```
#include "interrupts.h"
#include "definitions.h"
```

Include dependency graph for interrupts.c:


interrupts.c
interrupts.h
definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

Functions

- void Sys_Start_AtomicSection ()
- void Sys_End_AtomicSection ()

8.4.1 Detailed Description

includes basic system calls to create atomic sections. (Sections that cannot be interrupted)

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

2015

Copyright

> adapted FreeBSD License (see http://openswarm.org/license)

To protect sections of code from any interruptions one has to use the following code:

```
// do something
Sys_Start_AtomicSection();
    //do something which should not be interrupted
Sys_End_AtomicSection();
// do something else
```

8.4.2 Function Documentation

8.4.2.1 void Sys_End_AtomicSection (void) [inline]

Starts an atomic section

This Function starts an atomic section. This means the code afterwards cannot be interrupted by any interrupt.

Precondition

> Sys_Start_AtomicSection() must have been called.

Warning

> Do not execute Sys_End_AtomicSection() without having called Sys_Start_AtomicSection() once. Otherwise, the interrupt priority will be set to SYS_IRQP_SYSTEM_TIMER. This might cause errors.

Definition at line 58 of file interrupts.c.

8.4.2.2 void Sys_Start_AtomicSection (void) [inline]

Starts an atomic section

This Function starts an atomic section. This means the code afterwards cannot be interrupted by any interrupt.

Postcondition

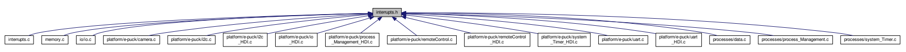
> Sys_End_AtomicSection() must be called to execute any interrupt that happened or will happen.

Definition at line 42 of file interrupts.c.

8.5 interrupts.h File Reference

includes system calls to create atomic sections. (Sections that cannot be interrupted)

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


interrupts.h
interrupts.c
memory.c
io/io.c
platform/e-puck/camera.c
platform/e-puck/i2c.c
platform/e-puck/i2c_HDI.c
platform/e-puck/io_HDI.c
platform/e-puck/process_Management_HDI.c
platform/e-puck/remoteControl.c
platform/e-puck/remoteControl_HDI.c
platform/e-puck/system_Timer_HDI.c
platform/e-puck/uart.c
platform/e-puck/uart_HDI.c
processes/data.c
processes/process_Management.c
processes/system_Timer.c

Macros

- #define SYS_IRQP_MAX 7
- #define SYS_IRQP_SYSTEM_TIMER 2
- #define SYS_IRQP_IO_TIMER 3
- #define SYS_IRQP_UART1 4
- #define SYS_IRQP_UART2 4
- #define SYS_IRQP_I2C 5
- #define SYS_IRQP_REMOTECONTROL 4
- #define SYS_IRQP_CAMERA_PIXEL 5
- #define SYS_IRQP_CAMERA_LINE 6
- #define SYS_IRQP_CAMERA_FRAME 7

Functions

- void Sys_Start_AtomicSection (void)
- void Sys_End_AtomicSection (void)

8.5.1 Detailed Description

includes system calls to create atomic sections. (Sections that cannot be interrupted)

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

{03 September 2015}

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.5.2 Macro Definition Documentation

8.5.2.1 #define SYS_IRQP_CAMERA_FRAME 7

interrupt priority for the camera frame interrupt

Definition at line 35 of file interrupts.h.

8.5.2.2 #define SYS_IRQP_CAMERA_LINE 6

interrupt priority for the camera line interrupt

Definition at line 34 of file interrupts.h.

8.5.2.3 #define SYS_IRQP_CAMERA_PIXEL 5

interrupt priority for the camera pixel interrupt

Definition at line 33 of file interrupts.h.

8.5.2.4 #define SYS_IRQP_I2C 5

interrupt priority for the I2C interrupt

Definition at line 29 of file interrupts.h.

8.5.2.5 #define SYS_IRQP_IO_TIMER 3

interrupt priority for the I/O timer interrupt

Definition at line 24 of file interrupts.h.

8.5.2.6 #define SYS_IRQP_MAX 7

maximum interrupt priority

Definition at line 20 of file interrupts.h.

8.5.2.7 #define SYS_IRQP_REMOTECONTROL 4

interrupt priority for the remote control interrupt

Definition at line 31 of file interrupts.h.

8.5.2.8 #define SYS_IRQP_SYSTEM_TIMER 2

interrupt priority for the system timer interrupt

Definition at line 22 of file interrupts.h.

8.5.2.9 #define SYS_IRQP_UART1 4

interrupt priority for the UART1 interrupt

Definition at line 26 of file interrupts.h.

8.5.2.10 #define SYS_IRQP_UART2 4

interrupt priority for the UART2 interrupt

Definition at line 27 of file interrupts.h.

8.5.3 Function Documentation

8.5.3.1 void Sys_End_AtomicSection (void) `[inline]`

Starts an atomic section

This Function starts an atomic section. This means the code afterwards cannot be interrupted by any interrupt.

Precondition

Sys_Start_AtomicSection() must have been called.

Warning

Do not execute Sys_End_AtomicSection() without having called Sys_Start_AtomicSection() once. Otherwise, the interrupt priority will be set to SYS_IRQP_SYSTEM_TIMER. This might cause errors.

Definition at line 58 of file interrupts.c.

8.5.3.2 void Sys_Start_AtomicSection (void) `[inline]`

Starts an atomic section

This Function starts an atomic section. This means the code afterwards cannot be interrupted by any interrupt.

Postcondition

Sys_End_AtomicSection() must be called to execute any interrupt that happened or will happen.

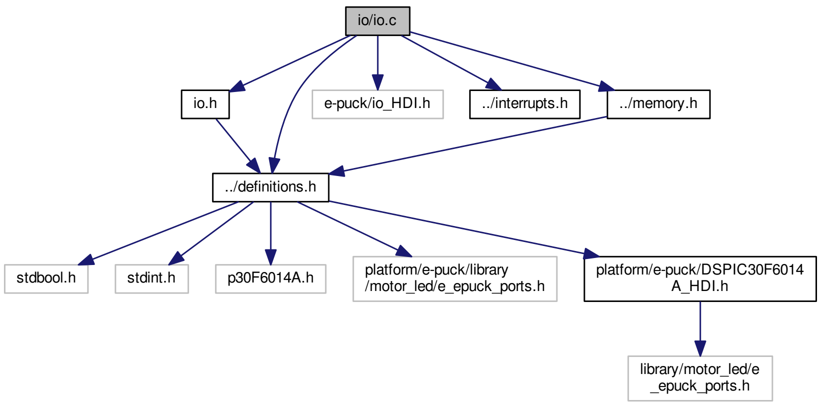
Definition at line 42 of file interrupts.c.

8.6 io/io.c File Reference

This file includes the IO timer to start and stop the timer. This timer executes IO functions periodically.

```
#include "io.h"
#include "../definitions.h"
#include "e-puck/io_HDI.h"
#include "../interrupts.h"
#include "../memory.h"
```

Include dependency graph for io.c:



Functions

- void Sys_Init_IOManagement (void)
- void Sys_Start_IOManagement (void)

- void Sys_Stop_IOManagement (void)
- void Sys_Stop_IOTimer ()
- void Sys_Continue_IOTimer ()
- void Sys_Reset_IOTimer ()
- void Sys_Disable_IOTimerInterrupt ()
- void Sys_Enable_IOTimerInterrupt ()
- void Sys_Force_IOTimerInterrupt ()
- bool Sys_Register_IOHandler (pFunction func)
- void Sys_Unregister_IOHandler (pFunction func)

8.6.1 Detailed Description

This file includes the IO timer to start and stop the timer. This timer executes IO functions periodically.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.6.2 Function Documentation

8.6.2.1 void Sys_Continue_IOTimer (void) `[inline]`

Continues the I/O Timer

This function continues the I/O Timer. Note that the timer continues to count where it stops.

Definition at line 71 of file io.c.

8.6.2.2 void Sys_Disable_IOTimerInterrupt (void) `[inline]`

Disables the I/O Timer

This function disables the I/O Timer interrupt. Note that the timer still continues to count.

Definition at line 91 of file io.c.

8.6.2.3 void Sys_Enable_IOTimerInterrupt (void) `[inline]`

Enables the I/O Timer

This function enables the I/O Timer interrupt.

Definition at line 101 of file io.c.

8.6.2.4 void Sys_Force_IOTimerInterrupt (void) [inline]

Force the I/O Timer interrupt.

This function forces a new I/O Timer interrupt even if the timer hasn't reached its threshold.

Definition at line 111 of file io.c.

8.6.2.5 void Sys_Init_IOManagement (void) [inline]

Initialises the I/O Management

This function initialises the I/O Timer and therefore the I/O Management.

Definition at line 31 of file io.c.

8.6.2.6 bool Sys_Register_IOHandler (pFunction *func*)

registers new I/O handler.

This function registers a new I/O handler which is executed every time the I/O timer interrupt occurs.

Parameters

func	pointer to the function that should be executed by the I/O timer periodically

Returns

bool was it successful?

Definition at line 123 of file io.c.

8.6.2.7 void Sys_Reset_IOTimer (void) [inline]

resets the I/O Timer

This function sets the I/O Timer counter to 0 and the I/O timer needs the full time duration to throw the interrupt.

Definition at line 81 of file io.c.

8.6.2.8 void Sys_Start_IOManagement (void) [inline]

Starts the I/O Management

This function starts the I/O Timer and therefore the I/O Management.

Definition at line 41 of file io.c.

8.6.2.9 void Sys_Stop_IOManagement (void) [inline]

Stops the I/O Management

This function stops the I/O Timer and therefore the I/O Management.

Definition at line 51 of file io.c.

8.6.2.10 void Sys_Stop_IOTimer (void) [inline]

Stops the I/O Timer

This function stops the I/O Timer.

Definition at line 61 of file io.c.

8.6.2.11 void Sys_Unregister_IOHandler (pFunction *func*)

unregisters new I/O handler.

This function unregisters a I/O handler identified by its function address.

Parameters

func	pointer to the function that should be executed by the I/O timer periodically

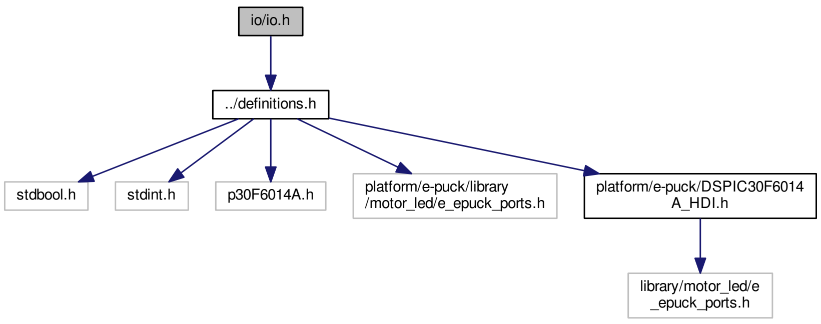
Definition at line 158 of file io.c.

8.7 io/io.h File Reference

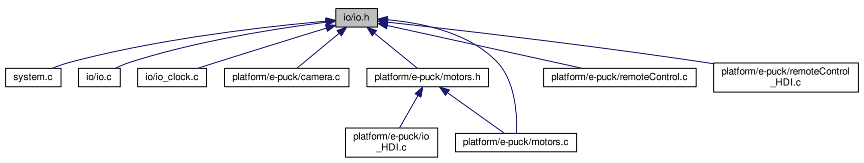
This file includes the IO timer to start and stop the timer. This timer executes IO functions periodically.

`#include "../definitions.h"`

Include dependency graph for io.h:


io/io.h
../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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



Functions

- void Sys_Init_IOManagement (void)
- void Sys_Start_IOManagement (void)
- void Sys_Stop_IOManagement (void)
- void Sys_Stop_IOTimer (void)
- void Sys_Continue_IOTimer (void)
- void Sys_Reset_IOTimer (void)
- void Sys_Disable_IOTimerInterrupt (void)

- void Sys_Enable_IOTimerInterrupt (void)
- void Sys_Force_IOTimerInterrupt (void)
- bool Sys_Register_IOHandler (pFunction func)
- void Sys_Unregister_IOHandler (pFunction func)

8.7.1 Detailed Description

This file includes the IO timer to start and stop the timer. This timer executes IO functions periodically.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

28 July 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.7.2 Function Documentation

8.7.2.1 void Sys_Continue_IOTimer (void) `[inline]`

Continues the I/O Timer

This function continues the I/O Timer. Note that the timer continues to count where it stops.

Definition at line 71 of file io.c.

8.7.2.2 void Sys_Disable_IOTimerInterrupt (void) `[inline]`

Disables the I/O Timer

This function disables the I/O Timer interrupt. Note that the timer still continues to count.

Definition at line 91 of file io.c.

8.7.2.3 void Sys_Enable_IOTimerInterrupt (void) `[inline]`

Enables the I/O Timer

This function enables the I/O Timer interrupt.

Definition at line 101 of file io.c.

8.7.2.4 void Sys_Force_IOTimerInterrupt (void) `[inline]`

Force the I/O Timer interrupt.

This function forces a new I/O Timer interrupt even if the timer hasn't reached its threshold.

Definition at line 111 of file io.c.

8.7.2.5 void Sys_Init_IOManagement (void) [inline]

Initialises the I/O Management

This function initialises the I/O Timer and therefore the I/O Management.

Definition at line 31 of file io.c.

8.7.2.6 bool Sys_Register_IOHandler (pFunction *func*)

registers new I/O handler.

This function registers a new I/O handler which is executed every time the I/O timer interrupt occurs.

Parameters

func	pointer to the function that should be executed by the I/O timer periodically

Returns

bool was it successful?

Definition at line 123 of file io.c.

8.7.2.7 void Sys_Reset_IOTimer (void) [inline]

resets the I/O Timer

This function sets the I/O Timer counter to 0 and the I/O timer needs the full time duration to throw the interrupt.

Definition at line 81 of file io.c.

8.7.2.8 void Sys_Start_IOManagement (void) [inline]

Starts the I/O Management

This function starts the I/O Timer and therefore the I/O Management.

Definition at line 41 of file io.c.

8.7.2.9 void Sys_Stop_IOManagement (void) [inline]

Stops the I/O Management

This function stops the I/O Timer and therefore the I/O Management.

Definition at line 51 of file io.c.

8.7.2.10 void Sys_Stop_IOTimer (void) [inline]

Stops the I/O Timer

This function stops the I/O Timer.

Definition at line 61 of file io.c.

8.7.2.11 void Sys_Unregister_IOHandler (pFunction *func*)

unregisters new I/O handler.

This function unregisters a I/O handler identified by its function address.

Parameters

func	pointer to the function that should be executed by the I/O timer periodically

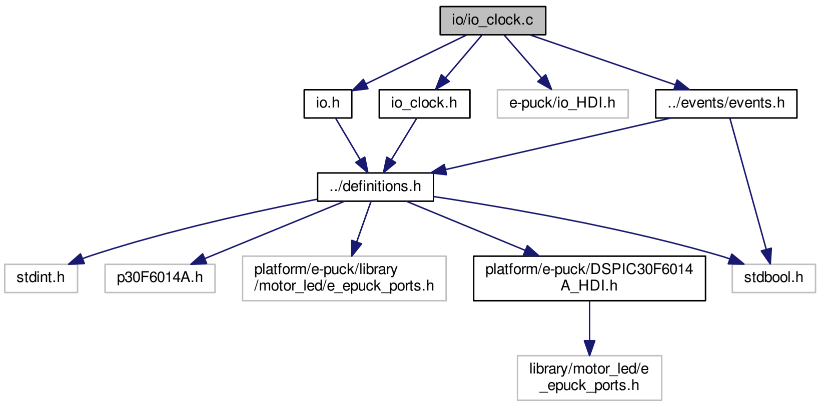
Definition at line 158 of file io.c.

8.8 io/io_clock.c File Reference

This file includes the system clock that can be used to measure time.

```
#include "io.h"
#include "io_clock.h"
#include "e-puck/io_HDI.h"
#include "../events/events.h"
```

Include dependency graph for io_clock.c:



Functions

- void Sys_SystemClock_Counter (void)
- void Sys_Init_Clock ()
- void Sys_Init_SystemTime ()
- uint32 Sys_Get_SystemTime ()
- uint32 Sys_Get_SystemClock ()

8.8.1 Detailed Description

This file includes the system clock that can be used to measure time.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

28 July 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.8.2 Function Documentation

8.8.2.1 uint32 Sys_Get_SystemClock (void) [inline]

returns the system clock/time in milliseconds

returns the system clock/time in milliseconds

Returns

uint32 time that has passed since OpenSwarm was started

Definition at line 82 of file io_clock.c.

8.8.2.2 uint32 Sys_Get_SystemTime (void) [inline]

Renaming of the function Sys_Get_SystemClock().

Renaming of the function Sys_Get_SystemClock().

Returns

uint32 time that has passed since OpenSwarm was started

Definition at line 71 of file io_clock.c.

8.8.2.3 void Sys_Init_Clock (void) [inline]

This function initialises the system clock

This function initialises the system clock which is in principle a counter that inicates passed milli seconds.

Definition at line 30 of file io_clock.c.

8.8.2.4 void Sys_Init_SystemTime (void) [inline]

Renaming of the function Sys_Init_Clock().

Renaming of the function Sys_Init_Clock().

Definition at line 41 of file io_clock.c.

8.8.2.5 void Sys_SystemClock_Counter ( )

calculates the system clock

This function calculates the system clock tick and increases the counter if a millisecond passed.

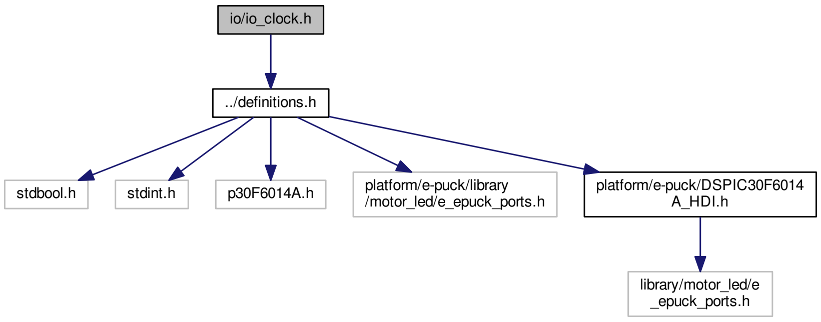
Definition at line 51 of file io_clock.c.

8.9 io/io_clock.h File Reference

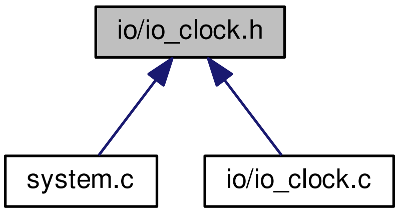
This file includes the system clock that can be used to measure time.

```
#include "../definitions.h"
```

Include dependency graph for io_clock.h:


io/io_clock.h
../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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



Functions

- void Sys_Init_Clock (void)
- void Sys_Init_SystemTime (void)
- uint32 Sys_Get_SystemTime (void)
- uint32 Sys_Get_SystemClock (void)

8.9.1 Detailed Description

This file includes the system clock that can be used to measure time.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

28 July 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.9.2 Function Documentation

8.9.2.1 uint32 Sys_Get_SystemClock (void) [inline]

returns the system clock/time in milliseconds

returns the system clock/time in milliseconds

Returns

uint32 time that has passed since OpenSwarm was started

Definition at line 82 of file io_clock.c.

8.9.2.2 uint32 Sys_Get_SystemTime (void) [inline]

Renaming of the function Sys_Get_SystemClock().

Renaming of the function Sys_Get_SystemClock().

Returns

uint32 time that has passed since OpenSwarm was started

Definition at line 71 of file io_clock.c.

8.9.2.3 void Sys_Init_Clock (void) [inline]

This function initialises the system clock

This function initialises the system clock which is in principle a counter that inicates passed milli seconds.

Definition at line 30 of file io_clock.c.

8.9.2.4 void Sys_Init_SystemTime (void) [inline]

Renaming of the function Sys_Init_Clock().

Renaming of the function Sys_Init_Clock().

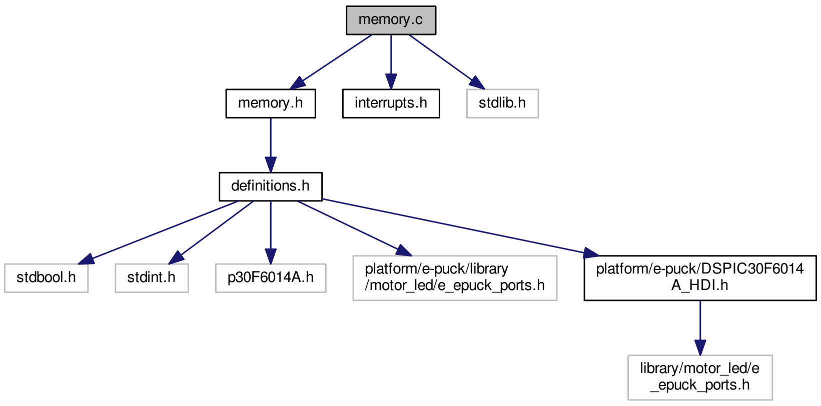
Definition at line 41 of file io_clock.c.

8.10 memory.c File Reference

includes functions to allocate, free, and copy memory

```
#include "memory.h"
#include "interrupts.h"
#include <stdlib.h>
```

Include dependency graph for memory.c:


memory.c
memory.h
interrupts.h
stdlib.h
definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

Functions

- void ∗ Sys_Malloc (uint16 length)
- void Sys_Free (void ∗data)
- void Sys_Memcpy (void ∗source_i, void ∗destination_o, uint16 length)

8.10.1 Detailed Description

includes functions to allocate, free, and copy memory

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

{05 September 2015}

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.10.2 Function Documentation

8.10.2.1 void Sys_Free (void $*$ *data*)

Function to free memory

This Function frees dynamic allocated memory. This freeing is performed as atomic action.

Parameters

data	pointer to memory that should be freed.

Definition at line 45 of file memory.c.

8.10.2.2 void∗ Sys_Malloc (uint16 *length*)

Function to allocate **length** bytes of memory

This Function allocates memory of the size **length**. This allocation is performed as atomic action.

Parameters

length	value how many bytes should be allocated

Returns

pointer to the allocated memory

Definition at line 26 of file memory.c.

8.10.2.3 void Sys_Memcpy (void ∗ *source_i,* void ∗ *destination_o,* uint16 *length*)

Function to copies memory of the size **length** from **source_i** to **destination_o**.

Function to copies memory of the size **length** from **source_i** to **destination_o**. This copying is performed as atomic action.

Parameters

source_i	pointer to the source
destination_o	pointer to the destination
length	size of the memory that has to be copied

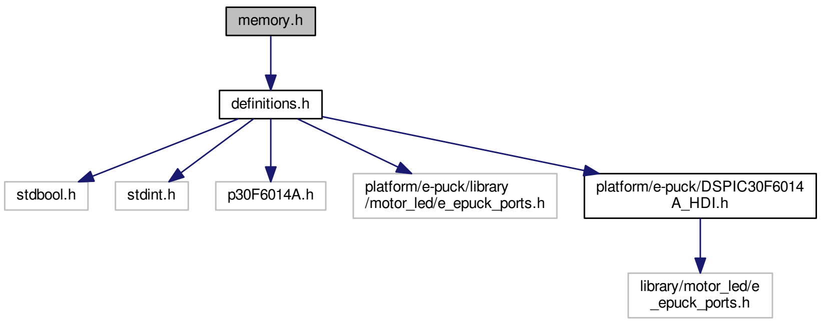
Definition at line 64 of file memory.c.

8.11 memory.h File Reference

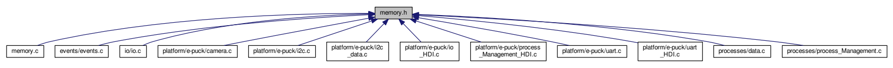
includes functions to allocate, free, and copy memory

```
#include "definitions.h"
```

Include dependency graph for memory.h:


memory.h
definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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


memory.h
memory.c
events/events.c
io/io.c
platform/e-puck/camera.c
platform/e-puck/i2c.c
platform/e-puck/i2c _data.c
platform/e-puck/io _HDI.c
platform/e-puck/process _Management_HDI.c
platform/e-puck/uart.c
platform/e-puck/uart _HDI.c
processes/data.c
processes/process_Management.c

Functions

- void ∗ Sys_Malloc (uint16 length)
- void Sys_Free (void ∗)
- void Sys_Memcpy (void ∗source, void ∗destination, uint16 length)

8.11.1 Detailed Description

includes functions to allocate, free, and copy memory

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

{05 September 2015}

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.11.2 Function Documentation

8.11.2.1 void Sys_Free (void ∗ *data*)

Function to free memory

This Function frees dynamic allocated memory. This freeing is performed as atomic action.

Parameters

data	pointer to memory that should be freed.

Definition at line 45 of file memory.c.

8.11.2.2 void∗ Sys_Malloc (uint16 *length*)

Function to allocate **length** bytes of memory

This Function allocates memory of the size **length**. This allocation is performed as atomic action.

Parameters

length	value how many bytes should be allocated

Returns

pointer to the allocated memory

Definition at line 26 of file memory.c.

8.11.2.3 void Sys_Memcpy (void ∗ *source_i,* void ∗ *destination_o,* uint16 *length*)

Function to copies memory of the size **length** from **source_i** to **destination_o**.

Function to copies memory of the size **length** from **source_i** to **destination_o**. This copying is performed as atomic action.

Parameters

source_i	pointer to the source
destination_o	pointer to the destination
length	size of the memory that has to be copied

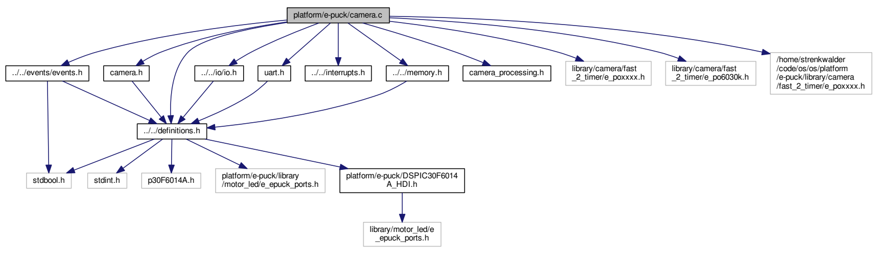
Definition at line 64 of file memory.c.

8.12 platform/e-puck/camera.c File Reference

This file includes functions to process data retrieved by a camera.

```
#include "camera.h"
#include "../../io/io.h"
#include "uart.h"
#include "../../definitions.h"
#include "../../events/events.h"
#include "../../interrupts.h"
#include "../../memory.h"
#include "camera_processing.h"
#include "library/camera/fast_2_timer/e_poxxxx.h"
#include "library/camera/fast_2_timer/e_po6030k.h"
```

Include dependency graph for camera.c:


platform/e-puck/camera.c
../../events/events.h
camera.h
../../io/io.h
uart.h
../../interrupts.h
../../memory.h
camera_processing.h
library/camera/fast _2_timer/e_poxxxx.h
library/camera/fast _2_timer/e_po6030k.h
/home/strenkwalder /code/os/os/platform /e-puck/library/camera /fast_2_timer/e_poxxxx.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

Macros

- #define FRAME_WIDTH 10
- #define FRAME_HEIGHT 10
- #define CAMERA_I2C_ADDRESS 0xDC

- #define RED_MAX 0x0C1C
- #define GREEN_MAX 0x189C
- #define BLUE_MAX 0x0C1C
- #define RED_THRESHOLD 0x060E
- #define GREEN_THRESHOLD 0x0E4E
- #define BLUE_THRESHOLD 0x060E
- #define CAM_WIDTH 160
- #define CAM_HEIGHT 160
- #define CAM_ZOOM_X 8
- #define CAM_ZOOM_Y 8
- #define CAM_W_SIZE 20
- #define CAM_H_SIZE 20
- #define CP_WI 120
- #define CP_RI 80
- #define CP_GI 80
- #define CP_BI 100
- #define COLOUR_THRESHOLD 766

Functions

- void Sys_Process_newPixel (void)
- void Sys_Process_newLine (void)
- void Sys_Process_newFrame (void)
- void Sys_Camera_PreProcessor (void)
- void Sys_Init_Camera ()
- void Sys_Start_Camera ()
- void Sys_Set_Preprocessing (pCameraPreProcessor func)

8.12.1 Detailed Description

This file includes functions to process data retrieved by a camera.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`
Yuri Kaszubowski Lopes `yurikazuba@gmail.com`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

Todo The used functions from the e-puck library are very time and computational intensive. These function can be rewritten to decrease the processing load.

8.12.2 Macro Definition Documentation

8.12.2.1 #define BLUE_MAX 0x0C1C

maximum value for received blue

Definition at line 44 of file camera.c.

8.12.2.2 #define BLUE_THRESHOLD 0x060E

threshold value for received blue

Definition at line 47 of file camera.c.

8.12.2.3 #define CAM_H_SIZE 20

post scale height frame

Definition at line 88 of file camera.c.

8.12.2.4 #define CAM_HEIGHT 160

height of the camera input frame

Definition at line 84 of file camera.c.

8.12.2.5 #define CAM_W_SIZE 20

post scale width frame

Definition at line 87 of file camera.c.

8.12.2.6 #define CAM_WIDTH 160

Initialises the I/O Management

This function initialises the I/O Timer and therefore the I/O Management.

param void return void

inline void Sys_Write_to_Camera(uint8 address, uint8∗ data, uint16 length){ uint8 ∗i2c_data = (uint8 ∗) Sys_↩ Malloc(length+1);

i2c_data[0] = address;

Sys_Memcpy(data, i2c_data+1,length);

Sys_I2C_SentBytes(CAMERA_I2C_ADDRESS, i2c_data, length+1); }width of the camera input frame

Definition at line 83 of file camera.c.

8.12.2.7 #define CAM_ZOOM_X 8

zoom factor to scale the frame

Definition at line 85 of file camera.c.

8.12.2.8 #define CAM_ZOOM_Y 8

zoom factor to scale the frame

Definition at line 86 of file camera.c.

8.12.2.9 #define CAMERA_I2C_ADDRESS 0xDC

I2C address of the camera

Definition at line 40 of file camera.c.

8.12.2.10 #define COLOUR_THRESHOLD 766

threshold to decide if a colour pixel has been measured

Definition at line 540 of file camera.c.

8.12.2.11 #define CP_BI 100

blue factor to process and calibrate the camera

Definition at line 539 of file camera.c.

8.12.2.12 #define CP_GI 80

green factor to process and calibrate the camera

Definition at line 538 of file camera.c.

8.12.2.13 #define CP_RI 80

red factor to process and calibrate the camera

Definition at line 537 of file camera.c.

8.12.2.14 #define CP_WI 120

whitness factor to process and calibrate the camera

Definition at line 536 of file camera.c.

8.12.2.15 #define FRAME_HEIGHT 10

Height of the subframe of the image

Definition at line 39 of file camera.c.

8.12.2.16 #define FRAME_WIDTH 10

Width of the subframe of the image

Definition at line 38 of file camera.c.

8.12.2.17 #define GREEN_MAX 0x189C

maximum value for received green

Definition at line 43 of file camera.c.

8.12.2.18 #define GREEN_THRESHOLD 0x0E4E

threshold value for received green

Definition at line 46 of file camera.c.

8.12.2.19 #define RED_MAX 0x0C1C

maximum value for received red

Definition at line 42 of file camera.c.

8.12.2.20 #define RED_THRESHOLD 0x060E

threshold value for received red

Definition at line 45 of file camera.c.

8.12.3 Function Documentation

8.12.3.1 void Sys_Camera_PreProcessor (void)

processes an incoming camera frame and emits events according to used algorithm

This function processes an incoming camera frame and emits events according to used algorithm.

Todo rewrite the camera to computational less intensive functions

Definition at line 551 of file camera.c.

8.12.3.2 void Sys_Init_Camera (void)

Initialises the Camera

This function initialises the camera using e-puck library from Subversion at svn://svn.gna.org/svn/e-puck/trunk

Todo rewrite the camera to computational less intensive functions

Definition at line 99 of file camera.c.

8.12.3.3 void Sys_Process_newFrame (void) `[inline]`

8.12.3.4 void Sys_Process_newLine (void) `[inline]`

8.12.3.5 void Sys_Process_newPixel (void) `[inline]`

8.12.3.6 void Sys_Set_Preprocessing (pCameraPreProcessor *func*)

Defines a preprocessor callback functions.

Defines a preprocessor callback functions to process the frame.

Parameters

in	*func*	camera preprocessor which computes events out of the raw image

Definition at line 319 of file camera.c.

8.12.3.7 void Sys_Start_Camera (void)

Starts the Camera

This function starts the capturing using e-puck library from Subversion at svn://svn.gna.org/svn/e-puck/trunk

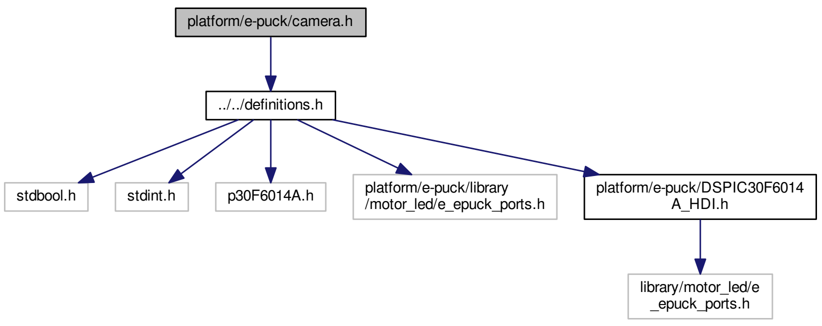
Todo rewrite the camera to computational less intensive functions

Definition at line 298 of file camera.c.

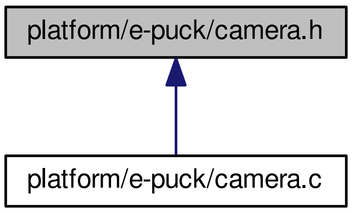
8.13 platform/e-puck/camera.h File Reference

This file includes functions to process data retrieved by a camera.

`#include "../../definitions.h"`
Include dependency graph for camera.h:


platform/e-puck/camera.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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



Data Structures

- struct sys_rgb_pixel_s

    This bitfield contains the structure of the received pixel of a camera.

Macros

- #define SYS_MAX_RED 0b00011111;
- #define SYS_MAX_GREEN 0b00111111;
- #define SYS_MAX_BLUE 0b00011111;

Typedefs

- typedef struct sys_rgb_pixel_s sys_rgb

    This bitfield contains the structure of the received pixel of a camera.
- typedef struct sys_rgb_pixel_s sys_rgb_pixel
- typedef void(∗ pCameraPreProcessor) (sys_rgb_pixel ∗∗frame, uint16 width, uint16 height)

Functions

- void Sys_Init_Camera (void)
- void Sys_Start_Camera (void)
- void Sys_Set_Preprocessing (pCameraPreProcessor func)
- sys_rgb_pixel ∗ getFinishedFrame ()
- bool isNewFrameAvailable ()

8.13.1 Detailed Description

This file includes functions to process data retrieved by a camera.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.13.2 Macro Definition Documentation

8.13.2.1 #define SYS_MAX_BLUE 0b00011111;

blue bits received

Definition at line 52 of file camera.h.

8.13.2.2 #define SYS_MAX_GREEN 0b00111111;

green bits received

Definition at line 51 of file camera.h.

8.13.2.3 #define SYS_MAX_RED 0b00011111;

red bits received

Definition at line 50 of file camera.h.

8.13.3 Typedef Documentation

8.13.3.1 typedef void(∗ pCameraPreProcessor) (sys_rgb_pixel ∗∗frame, uint16 width, uint16 height)

pointer to a camera preprocessor

Definition at line 63 of file camera.h.

8.13.3.2 typedef struct sys_rgb_pixel_s sys_rgb

This bitfield contains the structure of the received pixel of a camera.

8.13.3.3 typedef struct sys_rgb_pixel_s sys_rgb_pixel

8.13.4 Function Documentation

8.13.4.1 sys_rgb_pixel∗ getFinishedFrame ( )

8.13.4.2 bool isNewFrameAvailable ( )

8.13.4.3 void Sys_Init_Camera (void)

Initialises the Camera

This function initialises the camera using e-puck library from Subversion at svn://svn.gna.org/svn/e-puck/trunk

Todo rewrite the camera to computational less intensive functions

Definition at line 99 of file camera.c.

8.13.4.4 void Sys_Set_Preprocessing (pCameraPreProcessor *func*)

Defines a preprocessor callback functions.

Defines a preprocessor callback functions to process the frame.

Parameters

in	*func*	camera preprocessor which computes events out of the raw image

Definition at line 319 of file camera.c.

8.13.4.5 void Sys_Start_Camera (void)

Starts the Camera

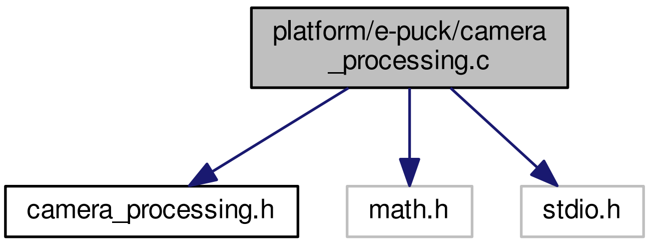
This function starts the capturing using e-puck library from Subversion at svn://svn.gna.org/svn/e-puck/trunk

Todo rewrite the camera to computational less intensive functions

Definition at line 298 of file camera.c.

8.14 platform/e-puck/camera_processing.c File Reference

```
#include "camera_processing.h"
#include <math.h>
#include <stdio.h>
```
Include dependency graph for camera_processing.c:



Macros

- #define CP_WI 100
- #define CP_WGB_I 80
- #define CP_RI 80
- #define CP_GI 40
- #define CP_BI 100
- #define CBP_WI 16
- #define CBP_RI 11
- #define CBP_GI 11
- #define CBP_BI 13
- #define CBP_DI 2

Functions

- void convertRGB565ToRGB888 (unsigned char rgb565[ ], unsigned char rgb888[ ])
- void getRGB565at (char ∗buffer, unsigned char rgb585[ ], int x, int y)
- void getRGB888at (char ∗buffer, unsigned char rgb888[ ], int x, int y)
- char nearestNeighborRGB (unsigned char ∗rbg888, char flag)
- char brushedColorFromRGB565 (unsigned char rgb565[ ], char flag)
- char getBrushedColorAt (char ∗buffer, char flag, int x, int y, int w)

Variables

- const unsigned char colorPositions [8][4]
- const int powerTbl [33] = {0,1,4,9,16,25,36,49,64,81,100,121,144,169,196,225,256,289,324,361,400,441,484,529,576,625,676
- const unsigned char colorBrushedPositions [8][4]

8.14.1 Detailed Description

Author

Yuri Kaszubowski Lopes yurikazuba@gmail.com

Version

1.0

Date

2014

8.14.2 Macro Definition Documentation

8.14.2.1 #define CBP_BI 13

Definition at line 74 of file camera_processing.c.

8.14.2.2 #define CBP_DI 2

Definition at line 75 of file camera_processing.c.

8.14.2.3 #define CBP_GI 11

Definition at line 73 of file camera_processing.c.

8.14.2.4 #define CBP_RI 11

Definition at line 72 of file camera_processing.c.

8.14.2.5 #define CBP_WI 16

Definition at line 71 of file camera_processing.c.

8.14.2.6 #define CP_BI 100

Definition at line 38 of file camera_processing.c.

8.14.2.7 #define CP_GI 40

Definition at line 37 of file camera_processing.c.

8.14.2.8 #define CP_RI 80

Definition at line 36 of file camera_processing.c.

8.14.2.9 #define CP_WGB_I 80

Definition at line 35 of file camera_processing.c.

8.14.2.10 #define CP_WI 100

Definition at line 34 of file camera_processing.c.

8.14.3 Function Documentation

8.14.3.1 char brushedColorFromRGB565 (unsigned char *rgb565[ ],* char *flag*)

Definition at line 88 of file camera_processing.c.

8.14.3.2 void convertRGB565ToRGB888 (unsigned char *rgb565[ ],* unsigned char *rgb888[ ]*)

Definition at line 17 of file camera_processing.c.

8.14.3.3 char getBrushedColorAt (char ∗ *buffer,* char *flag,* int *x,* int *y,* int *w*)

Definition at line 111 of file camera_processing.c.

8.14.3.4 void getRGB565at (char ∗ *buffer,* unsigned char *rgb585[ ],* int *x,* int *y*)

Definition at line 23 of file camera_processing.c.

8.14.3.5 void getRGB888at (char ∗ *buffer,* unsigned char *rgb888[ ],* int *x,* int *y*)

Definition at line 28 of file camera_processing.c.

8.14.3.6 char nearestNeighborRGB (unsigned char ∗ *rbg888,* char *flag*)

Definition at line 52 of file camera_processing.c.

8.14.4 Variable Documentation

8.14.4.1 const unsigned char colorBrushedPositions[8][4]

Initial value:

```
= {
    { CBP_DI, CBP_DI, CBP_DI, 'd' },
    { CBP_DI, CBP_GI, CBP_BI, 'c' },
    { CBP_RI, CBP_DI, CBP_BI, 'm' },
    { CBP_RI, CBP_GI, CBP_DI, 'y' },
    { CBP_DI, CBP_DI, CBP_BI, 'b' },
    { CBP_DI, CBP_GI, CBP_DI, 'g' },
    { CBP_RI, CBP_DI, CBP_DI, 'r' },
    { CBP_WI, CBP_WI, CBP_WI, 'w' }

}
```

Definition at line 76 of file camera_processing.c.

8.14.4.2 const unsigned char colorPositions[8][4]

Initial value:

```
= {
    { 0    , 0    , 0    , 'd' },
    { 0    , CP_GI, CP_BI, 'c' },
    { CP_RI, 0    , CP_BI, 'm' },
    { CP_RI, CP_GI, 0    , 'y' },
    { 0    , 0    , CP_BI, 'b' },
    { 0    , CP_GI, 0    , 'g' },
    { CP_RI, 0    , 0    , 'r' },
    { CP_WI, CP_WGB_I, CP_WGB_I, 'w' }

}
```

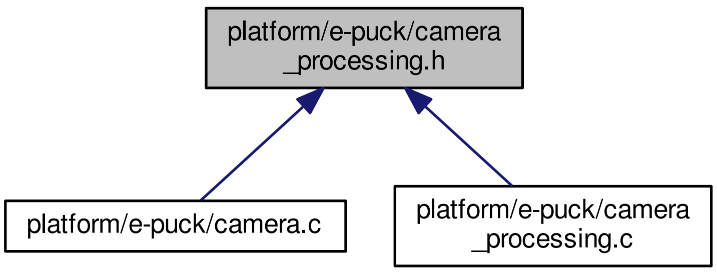
Definition at line 39 of file camera_processing.c.

8.14.4.3 const int powerTbl[33] = {0,1,4,9,16,25,36,49,64,81,100,121,144,169,196,225,256,289,324,361,400,441,484,529,576,625,676,729,784,841,900,961,

Definition at line 70 of file camera_processing.c.

8.15 platform/e-puck/camera_processing.h File Reference

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


platform/e-puck/camera_processing.h
platform/e-puck/camera.c
platform/e-puck/camera_processing.c

Functions

- void convertRGB565ToRGB888 (unsigned char rgb565[ ], unsigned char rgb888[ ])
- void getRGB565at (char ∗buffer, unsigned char rgb585[ ], int x, int y)
- void getRGB888at (char ∗buffer, unsigned char rgb888[ ], int x, int y)
- char nearestNeighborRGB (unsigned char ∗rbg888, char flag)
- char brushedColorFromRGB565 (unsigned char rgb565[ ], char flag)
- char getBrushedColorAt (char ∗buffer, char flag, int x, int y, int w)

8.15.1 Detailed Description

Author

Yuri Kaszubowski Lopes yurikazuba@gmail.com

Version

1.0

Date

2014

8.15.2 Function Documentation

8.15.2.1 char brushedColorFromRGB565 (unsigned char *rgb565[ ],* char *flag*)

Definition at line 88 of file camera_processing.c.

8.15.2.2 void convertRGB565ToRGB888 (unsigned char *rgb565[ ],* unsigned char *rgb888[ ]*)

Definition at line 17 of file camera_processing.c.

8.15.2.3 char getBrushedColorAt (char ∗ *buffer,* char *flag,* int *x,* int *y,* int *w*)

Definition at line 111 of file camera_processing.c.

8.15.2.4 void getRGB565at (char ∗ *buffer,* unsigned char *rgb585[ ],* int *x,* int *y*)

Definition at line 23 of file camera_processing.c.

8.15.2.5 void getRGB888at (char ∗ *buffer,* unsigned char *rgb888[ ],* int *x,* int *y*)

Definition at line 28 of file camera_processing.c.

8.15.2.6 char nearestNeighborRGB (unsigned char ∗ *rbg888,* char *flag*)

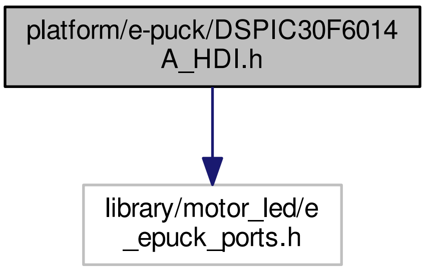
Definition at line 52 of file camera_processing.c.

8.16 platform/e-puck/DSPIC30F6014A_HDI.h File Reference

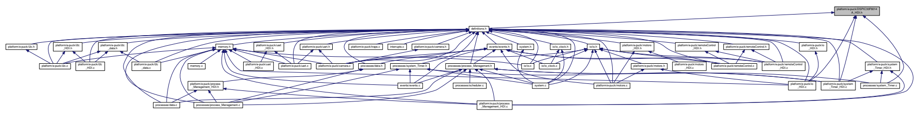
Hardware dependent declarations and definitions.

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

Include dependency graph for DSPIC30F6014A_HDI.h:



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



Macros

- #define ADDRESS_IVT 0x000004
- #define ADDRESS_ITV_OSC_FAIL ADDRESS_IVT+2
- #define ADDRESS_ITV_ADDRESS_ERROR ADDRESS_IVT+4
- #define ADDRESS_ITV_STACK_ERROR ADDRESS_IVT+6
- #define ADDRESS_ITV_MATH_ERROR ADDRESS_IVT+8
- #define ADDRESS_IVT_T1 0x00001A
- #define ADDRESS_AIVT 0x000084
- #define ADDRESS_AITV_OSC_FAIL ADDRESS_AIVT+2
- #define ADDRESS_AITV_ADDRESS_ERROR ADDRESS_AIVT+4
- #define ADDRESS_AITV_STACK_ERROR ADDRESS_AIVT+6
- #define ADDRESS_AITV_MATH_ERROR ADDRESS_AIVT+8
- #define ADDRESS_AIVT_T1 0x00009A

8.16.1 Detailed Description

Hardware dependent declarations and definitions.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

07 July 2014

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.16.2 Macro Definition Documentation

8.16.2.1 #define ADDRESS_AITV_ADDRESS_ERROR ADDRESS_AIVT+4

Definition at line 74 of file DSPIC30F6014A_HDI.h.

8.16.2.2 #define ADDRESS_AITV_MATH_ERROR ADDRESS_AIVT+8

Definition at line 76 of file DSPIC30F6014A_HDI.h.

8.16.2.3 #define ADDRESS_AITV_OSC_FAIL ADDRESS_AIVT+2

Definition at line 73 of file DSPIC30F6014A_HDI.h.

8.16.2.4 #define ADDRESS_AITV_STACK_ERROR ADDRESS_AIVT+6

Definition at line 75 of file DSPIC30F6014A_HDI.h.

8.16.2.5 #define ADDRESS_AIVT 0x000084

Definition at line 72 of file DSPIC30F6014A_HDI.h.

8.16.2.6 #define ADDRESS_AIVT_T1 0x00009A

Definition at line 77 of file DSPIC30F6014A_HDI.h.

8.16.2.7 #define ADDRESS_ITV_ADDRESS_ERROR ADDRESS_IVT+4

Definition at line 66 of file DSPIC30F6014A_HDI.h.

8.16.2.8 #define ADDRESS_ITV_MATH_ERROR ADDRESS_IVT+8

Definition at line 68 of file DSPIC30F6014A_HDI.h.

8.16.2.9 #define ADDRESS_ITV_OSC_FAIL ADDRESS_IVT+2

Definition at line 65 of file DSPIC30F6014A_HDI.h.

8.16.2.10 #define ADDRESS_ITV_STACK_ERROR ADDRESS_IVT+6

Definition at line 67 of file DSPIC30F6014A_HDI.h.

8.16.2.11 #define ADDRESS_IVT 0x000004

Definition at line 64 of file DSPIC30F6014A_HDI.h.

8.16.2.12 #define ADDRESS_IVT_T1 0x00001A

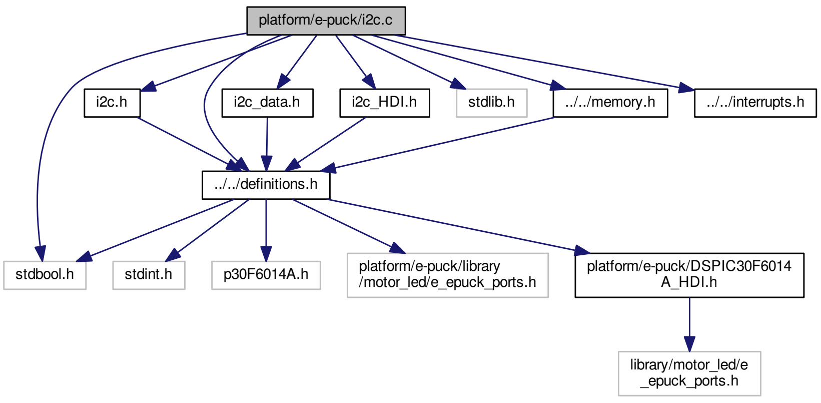
Definition at line 69 of file DSPIC30F6014A_HDI.h.

8.17 platform/e-puck/i2c.c File Reference

This file includes functions to read and write on the I2C interface.

```
#include "i2c.h"
#include "i2c_data.h"
#include "i2c_HDI.h"
#include <stdlib.h>
#include <stdbool.h>
#include "../../definitions.h"
#include "../../memory.h"
#include "../../interrupts.h"
```

Include dependency graph for i2c.c:



Functions

- void Sys_I2C_Send_Start ()
- void Sys_I2C_Send_Restart (void)
- void Sys_I2C_Send_Stop (void)
- void Sys_I2C_Send_ACK (void)
- void Sys_I2C_Send_NACK (void)
- void Sys_I2C_Start_Reading (void)
- char Sys_I2C_ReadByte (void)
- void Sys_I2C_WriteByte (uint8 byte)
- void Sys_Init_I2C ()
- void Sys_Start_I2C ()
- void Sys_Pause_I2C ()
- void Sys_Contine_I2C ()
- void Sys_Stop_I2C ()

- void Sys_I2C_SentBytes (uint8 address, uint8 ∗bytes, uint16 length)
- void Sys_I2C_Read (uint8 address, uint8 ∗intern_address, uint16 length, pByteFunction bytehandler)

8.17.1 Detailed Description

This file includes functions to read and write on the I2C interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.17.2 Function Documentation

8.17.2.1 void Sys_Contine_I2C (void) [inline]

continues the I2C interface

This function continues the I2C interface.

Definition at line 72 of file i2c.c.

8.17.2.2 void Sys_I2C_Read (uint8 *address*, uint8 ∗ *intern_address*, uint16 *length*, pByteFunction *bytehandler*)

reads bytes from an I2C device

This function first sends a reading request to the I2C device and, then, handles the incoming bytes with a callback function.

Parameters

in	*address*	The address of the I2C device that should receive the request
in	*intern_address*	A pointer to the address which should be read
in	*length*	the number of bytes of the address
in	*bytehandler*	a pointer to the handler function that processes the incoming bytes.

Definition at line 367 of file i2c.c.

8.17.2.3 char Sys_I2C_ReadByte ( ) [inline]

reads a byte via the I2C interface

This function reads a byte.

Definition at line 316 of file i2c.c.

8.17.2.4 void Sys_I2C_Send_ACK ( ) [inline]

sends a ack bits via the I2C interface

This function sends a ack bits.

Definition at line 286 of file i2c.c.

8.17.2.5 void Sys_I2C_Send_NACK ( ) [inline]

sends a nack bits via the I2C interface

This function sends a nack bits.

Definition at line 296 of file i2c.c.

8.17.2.6 void Sys_I2C_Send_Restart ( ) [inline]

sends a restart bits via the I2C interface

This function sends a restart bits.

Definition at line 266 of file i2c.c.

8.17.2.7 void Sys_I2C_Send_Start ( ) [inline]

sends a start bits via the I2C interface

This function sends a start bits.

Definition at line 256 of file i2c.c.

8.17.2.8 void Sys_I2C_Send_Stop ( ) [inline]

sends a stop bits via the I2C interface

This function sends a stop bits.

Definition at line 276 of file i2c.c.

8.17.2.9 void Sys_I2C_SentBytes (uint8 *address*, uint8 ∗ *bytes*, uint16 *length*)

adds bytes into a writing buffer

This function adds bytes into a writing buffer that are written as soon as the I2C is idle.

Note

all bytes are written in sequence

Parameters

in	*address*	The address of the I2C device that should receive the data
in	*bytes*	A pointer to the data which should be sent
in	*length*	the number of bytes to send

Definition at line 341 of file i2c.c.

8.17.2.10 void Sys_I2C_Start_Reading ( ) [inline]

sends a reading bits via the I2C interface

This function sends a reading bits.

Definition at line 306 of file i2c.c.

8.17.2.11 void Sys_I2C_WriteByte (uint8 *byte*) [inline]

writes a byte via the I2C interface

This function writes a byte.

Parameters

byte	the byte that has to be written

Definition at line 327 of file i2c.c.

8.17.2.12 void Sys_Init_I2C (void) [inline]

Initialises the I2C interface

This function initialises the I2C interface.

Definition at line 42 of file i2c.c.

8.17.2.13 void Sys_Pause_I2C (void) [inline]

pauses the I2C interface

This function pauses the I2C interface.

Definition at line 62 of file i2c.c.

8.17.2.14 void Sys_Start_I2C (void) [inline]

Starts the I2C interface

This function starts the I2C interface.

Definition at line 52 of file i2c.c.

8.17.2.15 void Sys_Stop_I2C (void) [inline]

stops the I2C interface

This function stops the I2C interface.

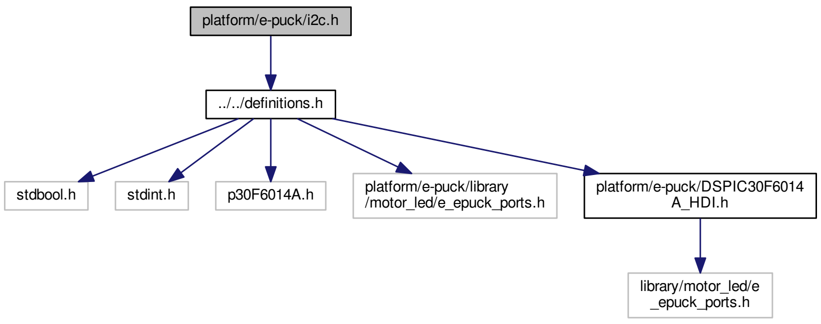
Definition at line 82 of file i2c.c.

8.18 platform/e-puck/i2c.h File Reference

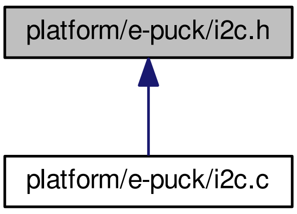
This file includes functions to read and write on the I2C interface.

```
#include "../../definitions.h"
```

Include dependency graph for i2c.h:


platform/e-puck/i2c.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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



Functions

- void Sys_Init_I2C (void)
- void Sys_Start_I2C (void)
- void Sys_Pause_I2C (void)
- void Sys_Contine_I2C (void)
- void Sys_Stop_I2C (void)
- void Sys_I2C_SentBytes (uint8 address, uint8 ∗bytes, uint16 length)
- void Sys_I2C_Read (uint8 address, uint8 ∗intern_address, uint16 length, pByteFunction bytehandler)

8.18.1 Detailed Description

This file includes functions to read and write on the I2C interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.18.2 Function Documentation

8.18.2.1 void Sys_Contine_I2C (void) `[inline]`

continues the I2C interface

This function continues the I2C interface.

Definition at line 72 of file i2c.c.

8.18.2.2 void Sys_I2C_Read (uint8 *address*, uint8 ∗ *intern_address*, uint16 *length*, pByteFunction *bytehandler*)

reads bytes from an I2C device

This function first sends a reading request to the I2C device and, then, handles the incoming bytes with a callback function.

Parameters

`in`	*address*	The address of the I2C device that should receive the request
`in`	*intern_address*	A pointer to the address which should be read
`in`	*length*	the number of bytes of the address
`in`	*bytehandler*	a pointer to the handler function that processes the incoming bytes.

Definition at line 367 of file i2c.c.

8.18.2.3 void Sys_I2C_SentBytes (uint8 *address*, uint8 ∗ *bytes*, uint16 *length*)

adds bytes into a writing buffer

This function adds bytes into a writing buffer that are written as soon as the I2C is idle.

Note

all bytes are written in sequence

Parameters

`in`	*address*	The address of the I2C device that should receive the data
`in`	*bytes*	A pointer to the data which should be sent
`in`	*length*	the number of bytes to send

Definition at line 341 of file i2c.c.

8.18.2.4 void Sys_Init_I2C (void) `[inline]`

Initialises the I2C interface

This function initialises the I2C interface.

Definition at line 42 of file i2c.c.

8.18.2.5 void Sys_Pause_I2C (void) `[inline]`

pauses the I2C interface

This function pauses the I2C interface.

Definition at line 62 of file i2c.c.

8.18.2.6 void Sys_Start_I2C (void) `[inline]`

Starts the I2C interface

This function starts the I2C interface.

Definition at line 52 of file i2c.c.

8.18.2.7 void Sys_Stop_I2C (void) `[inline]`

stops the I2C interface

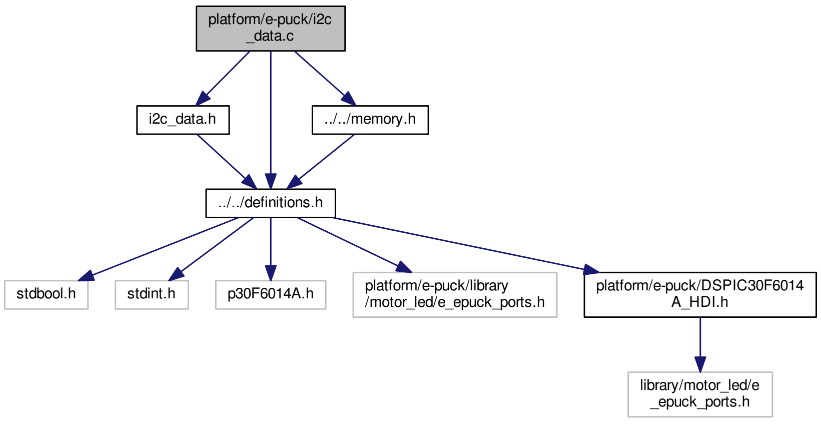
This function stops the I2C interface.

Definition at line 82 of file i2c.c.

8.19 platform/e-puck/i2c_data.c File Reference

This file includes functions to read and write on the I2C interface.

```
#include "i2c_data.h"
#include "../../definitions.h"
#include "../../memory.h"
```
Include dependency graph for i2c_data.c:


platform/e-puck/i2c_data.c
i2c_data.h
../../memory.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_I2C_RemoveOldestMessage (sys_i2c_messages ∗∗list)
- void Sys_I2C_FreeMessages (sys_i2c_messages ∗list)
- void Sys_I2C_AppendMessages (sys_i2c_msg ∗item)

Variables

- sys_i2c_messages ∗ sys_i2c_msgs = 0

8.19.1 Detailed Description

This file includes functions to read and write on the I2C interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.19.2 Function Documentation

8.19.2.1 void Sys_I2C_AppendMessages (**sys_i2c_msg** ∗ *item*)

appends an element to the linked list.

This function appends on the bottom of the linked list.

Parameters

`in,out`	*item*	pointer to a element that should be added

Definition at line 69 of file i2c_data.c.

8.19.2.2 void Sys_I2C_FreeMessages (**sys_i2c_messages** ∗ *list*)

frees all messages of the linked list

This function frees all messages of the linked list.

Parameters

`in`	*list*	pointer to a list of elements that should be removed

Definition at line 47 of file i2c_data.c.

8.19.2.3 void Sys_I2C_RemoveOldestMessage (**sys_i2c_messages** ∗∗ *list*)

removes oldest message from the linked list

This function removes the oldest message (first element) of the linked list

Parameters

`in,out`	*list*	pointer to the linked list

Definition at line 30 of file i2c_data.c.

8.19.3 Variable Documentation

8.19.3.1 sys_i2c_messages∗ sys_i2c_msgs = 0

Pointer to the linked list of messages

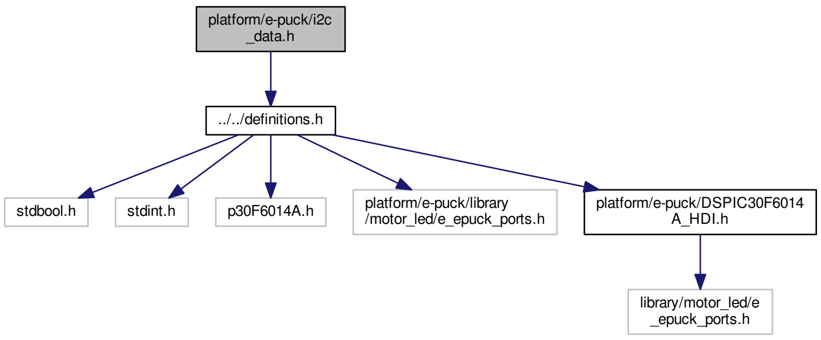
Definition at line 21 of file i2c_data.c.

8.20 platform/e-puck/i2c_data.h File Reference

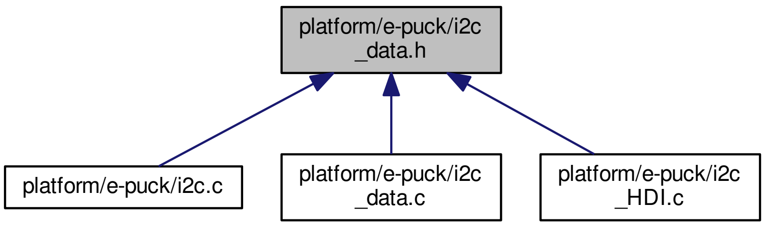
This file includes functions to read and write on the I2C interface.

`#include "../../definitions.h"`

Include dependency graph for i2c_data.h:


platform/e-puck/i2c _data.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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


platform/e-puck/i2c _data.h
platform/e-puck/i2c.c
platform/e-puck/i2c _data.c
platform/e-puck/i2c _HDI.c

Data Structures

- struct sys_i2c_message_s

Linked list element of messages that need to be sent via I2C.

Typedefs

- typedef struct sys_i2c_message_s sys_i2c_message

  Linked list element of messages that need to be sent via I2C.
- typedef struct sys_i2c_message_s sys_i2c_messages
- typedef struct sys_i2c_message_s sys_i2c_msg

Enumerations

- enum sys_I2C_state {
  I2C_IDLE = 0, I2C_IS_STARTING, I2C_STARTED, I2C_IS_READING,
  I2C_IS_SENDING, I2C_SENT, I2C_ACKNOWLEDGED, I2C_IS_STOPPING,
  I2C_ERROR }
- enum sys_I2C_mode {
  I2C_IDLE_MODE = 0, I2C_WRITING_ADDRESS_MODE, I2C_READING_BYTES_MODE, I2C_WRITIN↩
  G_BYTES_MODE,
  I2C_ERROR_MODE }

Functions

- void Sys_I2C_AppendMessages (sys_i2c_msg ∗item)
- void Sys_I2C_RemoveOldestMessage (sys_i2c_messages ∗∗list)
- void Sys_I2C_FreeMessages (sys_i2c_messages ∗list)

Variables

- sys_i2c_messages ∗ sys_i2c_msgs

8.20.1 Detailed Description

This file includes functions to read and write on the I2C interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.20.2 Typedef Documentation

8.20.2.1 typedef struct sys_i2c_message_s sys_i2c_message

Linked list element of messages that need to be sent via I2C.

8.20.2.2 typedef struct sys_i2c_message_s sys_i2c_messages

8.20.2.3 typedef struct sys_i2c_message_s sys_i2c_msg

8.20.3 Enumeration Type Documentation

8.20.3.1 enum sys_I2C_mode

Enumerator

I2C_IDLE_MODE

I2C_WRITING_ADDRESS_MODE

I2C_READING_BYTES_MODE

I2C_WRITING_BYTES_MODE

I2C_ERROR_MODE

Definition at line 25 of file i2c_data.h.

8.20.3.2 enum sys_I2C_state

Enumerator

I2C_IDLE

I2C_IS_STARTING

I2C_STARTED

I2C_IS_READING

I2C_IS_SENDING

I2C_SENT

I2C_ACKNOWLEDGED

I2C_IS_STOPPING

I2C_ERROR

Definition at line 24 of file i2c_data.h.

8.20.4 Function Documentation

8.20.4.1 void Sys_I2C_AppendMessages (sys_i2c_msg ∗ *item*)

appends an element to the linked list.

This function appends on the bottom of the linked list.

Parameters

in,out	*item*	pointer to a element that should be added

Definition at line 69 of file i2c_data.c.

8.20.4.2 void Sys_I2C_FreeMessages (sys_i2c_messages ∗ *list*)

frees all messages of the linked list

This function frees all messages of the linked list.

Parameters

in	*list*	pointer to a list of elements that should be removed

Definition at line 47 of file i2c_data.c.

8.20.4.3 void Sys_I2C_RemoveOldestMessage (sys_i2c_messages ∗∗ *list*)

removes oldest message from the linked list

This function removes the oldest message (first element) of the linked list

Parameters

in,out	*list*	pointer to the linked list

Definition at line 30 of file i2c_data.c.

8.20.5 Variable Documentation

8.20.5.1 sys_i2c_messages∗ sys_i2c_msgs

Pointer to the linked list of messages

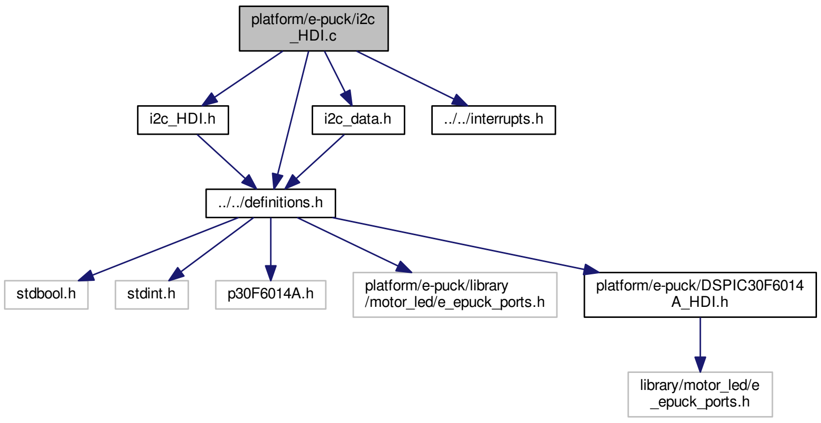
Definition at line 21 of file i2c_data.c.

8.21 platform/e-puck/i2c_HDI.c File Reference

Hardware dependent implementations to read and write on the I2C interface.

```
#include "i2c_HDI.h"
#include "i2c_data.h"
#include "../../definitions.h"
#include "../../interrupts.h"
```

Include dependency graph for i2c_HDI.c:



Functions

- void Sys_Init_I2C_HDI ()
- void Sys_Start_I2C_HDI (void)
- void Sys_Pause_I2C_HDI (void)
- void Sys_Contine_I2C_HDI (void)
- void Sys_Stop_I2C_HDI (void)
- void Sys_I2C_Send_Start_HDI ()
- void Sys_I2C_Send_Restart_HDI (void)
- void Sys_I2C_Send_Stop_HDI (void)
- void Sys_I2C_Send_ACK_HDI (void)
- void Sys_I2C_Send_NACK_HDI (void)
- void Sys_I2C_Start_Reading_HDI ()
- char Sys_I2C_ReadByte_HDI ()
- void Sys_I2C_WriteByte_HDI (uint8 byte)

8.21.1 Detailed Description

Hardware dependent implementations to read and write on the I2C interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.21.2 Function Documentation

8.21.2.1 void Sys_Contine_I2C_HDI (void) [inline]

continues the I2C interface

This function continues the I2C interface.

Definition at line 74 of file i2c_HDI.c.

8.21.2.2 char Sys_I2C_ReadByte_HDI (void) [inline]

reads a byte via the I2C interface

This function reads a byte.

Definition at line 178 of file i2c_HDI.c.

8.21.2.3 void Sys_I2C_Send_ACK_HDI (void) [inline]

sends a ack bits via the I2C interface

This function sends a ack bits.

Definition at line 130 of file i2c_HDI.c.

8.21.2.4 void Sys_I2C_Send_NACK_HDI (void) [inline]

sends a nack bits via the I2C interface

This function sends a nack bits.

Definition at line 146 of file i2c_HDI.c.

8.21.2.5 void Sys_I2C_Send_Restart_HDI (void) [inline]

sends a restart bits via the I2C interface

This function sends a restart bits.

Definition at line 108 of file i2c_HDI.c.

8.21.2.6 void Sys_I2C_Send_Start_HDI ( ) [inline]

sends a start bits via the I2C interface

This function sends a start bits.

Definition at line 96 of file i2c_HDI.c.

8.21.2.7 void Sys_I2C_Send_Stop_HDI (void) [inline]

sends a stop bits via the I2C interface

This function sends a stop bits.

Definition at line 120 of file i2c_HDI.c.

8.21.2.8 void Sys_I2C_Start_Reading_HDI (void) `[inline]`

sends a reading bits via the I2C interface

This function sends a reading bits.

Definition at line 162 of file i2c_HDI.c.

8.21.2.9 void Sys_I2C_WriteByte_HDI (uint8 *byte*) `[inline]`

writes a byte via the I2C interface

This function writes a byte.

Parameters

byte	the byte that has to be written

Definition at line 189 of file i2c_HDI.c.

8.21.2.10 void Sys_Init_I2C_HDI (void) `[inline]`

Initialises the I2C interface

This function initialises the I2C interface.

Definition at line 27 of file i2c_HDI.c.

8.21.2.11 void Sys_Pause_I2C_HDI (void) `[inline]`

pauses the I2C interface

This function pauses the I2C interface.

Definition at line 64 of file i2c_HDI.c.

8.21.2.12 void Sys_Start_I2C_HDI (void) `[inline]`

Starts the I2C interface

This function starts the I2C interface.

Definition at line 52 of file i2c_HDI.c.

8.21.2.13 void Sys_Stop_I2C_HDI (void) `[inline]`

stops the I2C interface

This function stops the I2C interface.

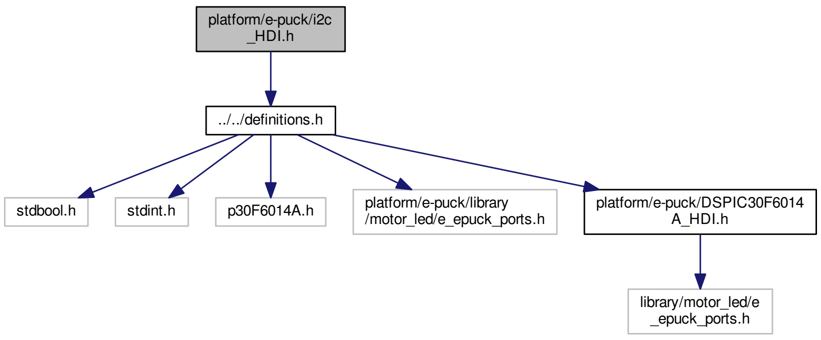
Definition at line 84 of file i2c_HDI.c.

8.22 platform/e-puck/i2c_HDI.h File Reference

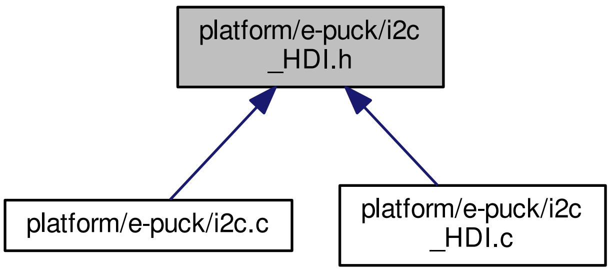
Hardware dependent implementations to read and write on the I2C interface.

```
#include "../../definitions.h"
```

Include dependency graph for i2c_HDI.h:


platform/e-puck/i2c _HDI.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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



Functions

- void Sys_I2C_Send_Start_HDI ()
- void Sys_I2C_Send_Restart_HDI (void)
- void Sys_I2C_Send_Stop_HDI (void)
- void Sys_I2C_Send_ACK_HDI (void)
- void Sys_I2C_Send_NACK_HDI (void)
- void Sys_I2C_Start_Reading_HDI (void)
- char Sys_I2C_ReadByte_HDI (void)
- void Sys_I2C_WriteByte_HDI (uint8 byte)
- void Sys_Init_I2C_HDI (void)
- void Sys_Start_I2C_HDI (void)
- void Sys_Pause_I2C_HDI (void)
- void Sys_Contine_I2C_HDI (void)
- void Sys_Stop_I2C_HDI (void)
- void Sys_I2C_SentBytes (uint8 address, uint8 ∗bytes, uint16 length)
- void Sys_I2C_Read (uint8 address, uint8 ∗intern_address, uint16 length, pByteFunction bytehandler)

8.22.1 Detailed Description

Hardware dependent implementations to read and write on the I2C interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.22.2 Function Documentation

8.22.2.1 void Sys_Contine_I2C_HDI (void) `[inline]`

continues the I2C interface

This function continues the I2C interface.

Definition at line 74 of file i2c_HDI.c.

8.22.2.2 void Sys_I2C_Read (**uint8** *address,* **uint8** ∗ *intern_address,* **uint16** *length,* **pByteFunction** *bytehandler*)

reads bytes from an I2C device

This function first sends a reading request to the I2C device and, then, handles the incoming bytes with a callback function.

Parameters

`in`	*address*	The address of the I2C device that should receive the request
`in`	*intern_address*	A pointer to the address which should be read
`in`	*length*	the number of bytes of the address
`in`	*bytehandler*	a pointer to the handler function that processes the incoming bytes.

Definition at line 367 of file i2c.c.

8.22.2.3 char Sys_I2C_ReadByte_HDI (void) `[inline]`

reads a byte via the I2C interface

This function reads a byte.

Definition at line 178 of file i2c_HDI.c.

8.22.2.4 void Sys_I2C_Send_ACK_HDI (void) `[inline]`

sends a ack bits via the I2C interface

This function sends a ack bits.

Definition at line 130 of file i2c_HDI.c.

8.22.2.5 void Sys_I2C_Send_NACK_HDI (void) [inline]

sends a nack bits via the I2C interface

This function sends a nack bits.

Definition at line 146 of file i2c_HDI.c.

8.22.2.6 void Sys_I2C_Send_Restart_HDI (void) [inline]

sends a restart bits via the I2C interface

This function sends a restart bits.

Definition at line 108 of file i2c_HDI.c.

8.22.2.7 void Sys_I2C_Send_Start_HDI ( ) [inline]

sends a start bits via the I2C interface

This function sends a start bits.

Definition at line 96 of file i2c_HDI.c.

8.22.2.8 void Sys_I2C_Send_Stop_HDI (void) [inline]

sends a stop bits via the I2C interface

This function sends a stop bits.

Definition at line 120 of file i2c_HDI.c.

8.22.2.9 void Sys_I2C_SentBytes (uint8 *address*, uint8 ∗ *bytes*, uint16 *length*)

adds bytes into a writing buffer

This function adds bytes into a writing buffer that are written as soon as the I2C is idle.

Note

all bytes are written in sequence

Parameters

in	*address*	The address of the I2C device that should receive the data
in	*bytes*	A pointer to the data which should be sent
in	*length*	the number of bytes to send

Definition at line 341 of file i2c.c.

8.22.2.10 void Sys_I2C_Start_Reading_HDI (void) [inline]

sends a reading bits via the I2C interface

This function sends a reading bits.

Definition at line 162 of file i2c_HDI.c.

8.22.2.11 void Sys_I2C_WriteByte_HDI (uint8 *byte*) [inline]

writes a byte via the I2C interface

This function writes a byte.

Parameters

byte	the byte that has to be written

Definition at line 189 of file i2c_HDI.c.

8.22.2.12 void Sys_Init_I2C_HDI (void) [inline]

Initialises the I2C interface

This function initialises the I2C interface.

Definition at line 27 of file i2c_HDI.c.

8.22.2.13 void Sys_Pause_I2C_HDI (void) [inline]

pauses the I2C interface

This function pauses the I2C interface.

Definition at line 64 of file i2c_HDI.c.

8.22.2.14 void Sys_Start_I2C_HDI (void) [inline]

Starts the I2C interface

This function starts the I2C interface.

Definition at line 52 of file i2c_HDI.c.

8.22.2.15 void Sys_Stop_I2C_HDI (void) [inline]

stops the I2C interface

This function stops the I2C interface.

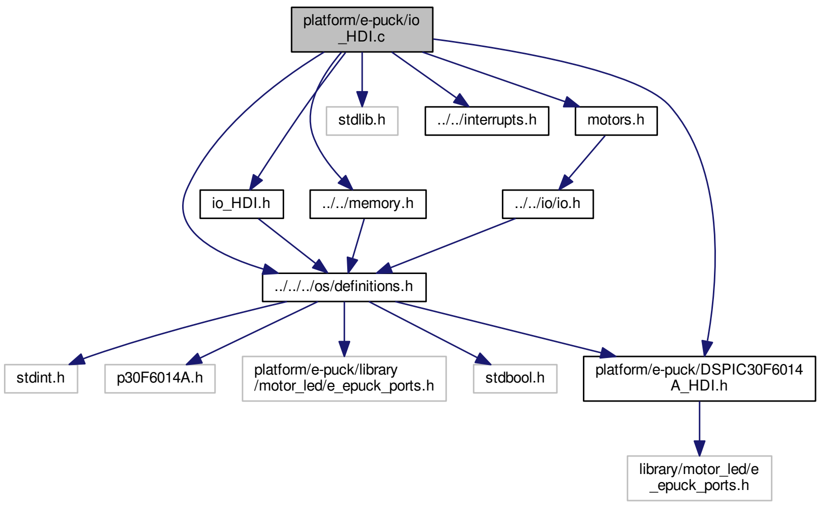
Definition at line 84 of file i2c_HDI.c.

8.23 platform/e-puck/io_HDI.c File Reference

Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.

```
#include "io_HDI.h"
#include <stdlib.h>
#include "DSPIC30F6014A_HDI.h"
#include "../../definitions.h"
#include "../../interrupts.h"
#include "../../memory.h"
#include "motors.h"
```

Include dependency graph for io_HDI.c:


platform/e-puck/io_HDI.c
stdlib.h
../../interrupts.h
motors.h
io_HDI.h
../../memory.h
../../io/io.h
../../../os/definitions.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
stdbool.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_Init_IOTimer_HDI ()
- void Sys_Start_IOTimer_HDI ()
- void Sys_Stop_IOTimer_HDI ()
- void Sys_Continue_IOTimer_HDI ()
- void Sys_Reset_IOTimer_HDI ()
- void __attribute__ ((interrupt, no_auto_psv))
- void Sys_Disable_IOTimerInterrupt_HDI ()
- void Sys_Enable_IOTimerInterrupt_HDI ()
- void Sys_Force_IOTimerInterrupt_HDI ()
- void Sys_IOTimer_code_HDI ()

Variables

- sys_periodical_IOHandler ∗ sys_iohandlers

8.23.1 Detailed Description

Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.23.2 Function Documentation

8.23.2.1 void __attribute__ ((interrupt, no_auto_psv))

Interrupt Service Routine for the Timer1 HDI

This Function starts the task-scheduling algorithm

Interrupt Service Routine for the Timer1 HDI (alternate)

This Function starts the task-scheduling algorithm

Interrupt Service Routine for the Timer2 HDI (alternate)

This Function starts the task-scheduling algorithm

Address error trap.

This function is called when an address error occurs. That means that a call address of a function or in the stack addresses an area outside the memory. Similarly, if a pointer points to memory outside the range, this trap happens.

Stack error trap.

This function is called when an stack error occurs. That means that the stack pointer, stack pointer limit, or frame pointer are pointing outside their range.

Math error trap.

This function is called when an math error occurs. That means an illegal math operation was performed (such as division by 0 or NaN).

Alternative Oscillator fail trap.

This function is called when an oscillator fail occurs. This should never happen.

Alternative address error trap.

This function is called when an address error occurs. That means that a call address of a function or in the stack addresses an area outside the memory. Similarly, if a pointer points to memory outside the range, this trap happens.

Alternative stack error trap.

This function is called when an stack error occurs. That means that the stack pointer, stack pointer limit, or frame pointer are pointing outside their range.

Alternative math error trap.

This function is called when an math error occurs. That means an illegal math operation was performed (such as division by 0 or NaN).

Default interrupt service routine.

This function is called when no other interrupt routine is specified.

Definition at line 110 of file io_HDI.c.

8.23.2.2 void Sys_Continue_IOTimer_HDI (void) `[inline]`

continues the I/O Timer

This function continues the I/O Timer.

Definition at line 86 of file io_HDI.c.

8.23.2.3 void Sys_Disable_IOTimerInterrupt_HDI (void) [inline]

Disables the Timer1 interrupt

Disables the Timer1 interrupt and sets the interrupt flag to 0

Definition at line 132 of file io_HDI.c.

8.23.2.4 void Sys_Enable_IOTimerInterrupt_HDI (void) [inline]

Enables the Timer1 interrupt

Enables the Timer1 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 143 of file io_HDI.c.

8.23.2.5 void Sys_Force_IOTimerInterrupt_HDI (void) [inline]

forces the Timer1 interrupt

forces the Timer1 interrupt to occur.

Definition at line 152 of file io_HDI.c.

8.23.2.6 void Sys_Init_IOTimer_HDI ( ) [inline]

initialises the I/O Timer

This function initialises the I/O Timer.

Definition at line 35 of file io_HDI.c.

8.23.2.7 void Sys_IOTimer_code_HDI ( ) [inline]

execution of all I/O handlers.

This function is executed every time the I/O timer is active and executes all I/O handlers

Definition at line 162 of file io_HDI.c.

8.23.2.8 void Sys_Reset_IOTimer_HDI (void) [inline]

resets the I/O Timer

This function resets the I/O Timer.

Definition at line 99 of file io_HDI.c.

8.23.2.9 void Sys_Start_IOTimer_HDI ( ) [inline]

starts the I/O Timer

This function starts the I/O Timer.

Definition at line 60 of file io_HDI.c.

8.23.2.10 void Sys_Stop_IOTimer_HDI (void) [inline]

stops the I/O Timer

This function stops the I/O Timer.

Definition at line 73 of file io_HDI.c.

8.23.3 Variable Documentation

8.23.3.1 sys_periodical_IOHandler∗ sys_iohandlers

List of I/O handlers

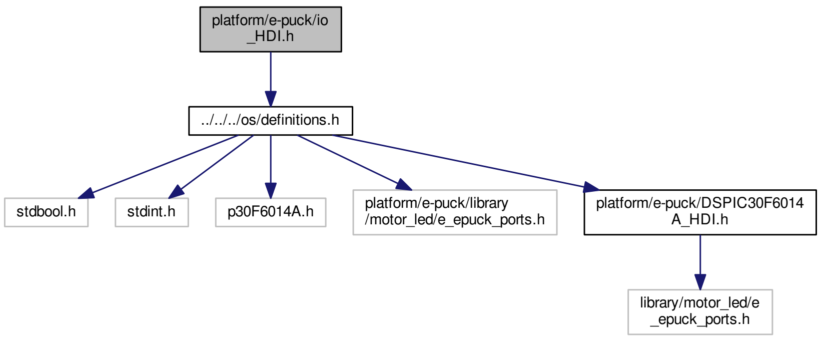
Definition at line 26 of file io_HDI.c.

8.24 platform/e-puck/io_HDI.h File Reference

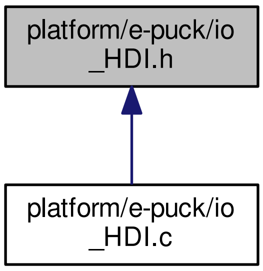
Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.

`#include "../../../os/definitions.h"`

Include dependency graph for io_HDI.h:


platform/e-puck/io_HDI.h
../../../os/definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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



Data Structures

- struct sys_periodical_IOHandler_s

Macros

- #define STEPS_PER_SECOND 10000
- #define STEPS_PER_MILISECOND 10

Typedefs

- typedef struct sys_periodical_IOHandler_s sys_periodical_IOHandler
- typedef struct sys_periodical_IOHandler_s sys_pIOHandler

Functions

- void Sys_Init_IOTimer_HDI ()
- void Sys_Start_IOTimer_HDI ()
- void Sys_IOTimer_code_HDI ()
- void Sys_Stop_IOTimer_HDI (void)
- void Sys_Continue_IOTimer_HDI (void)
- void Sys_Reset_IOTimer_HDI (void)
- void Sys_Disable_IOTimerInterrupt_HDI (void)
- void Sys_Enable_IOTimerInterrupt_HDI (void)
- void Sys_Force_IOTimerInterrupt_HDI (void)

Variables

- sys_periodical_IOHandler ∗ sys_iohandlers

8.24.1 Detailed Description

Hardware dependent implementations to start and stop the I/O timer. This timer executes IO functions periodically.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

10 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.24.2 Macro Definition Documentation

8.24.2.1 #define STEPS_PER_MILISECOND 10

Definition at line 26 of file io_HDI.h.

8.24.2.2 #define STEPS_PER_SECOND 10000

Definition at line 25 of file io_HDI.h.

8.24.3 Typedef Documentation

8.24.3.1 typedef struct sys_periodical_IOHandler_s sys_periodical_IOHandler

8.24.3.2 typedef struct sys_periodical_IOHandler_s sys_pIOHandler

8.24.4 Function Documentation

8.24.4.1 void Sys_Continue_IOTimer_HDI (void) [inline]

continues the I/O Timer

This function continues the I/O Timer.

Definition at line 86 of file io_HDI.c.

8.24.4.2 void Sys_Disable_IOTimerInterrupt_HDI (void) [inline]

Disables the Timer1 interrupt

Disables the Timer1 interrupt and sets the interrupt flag to 0

Definition at line 132 of file io_HDI.c.

8.24.4.3 void Sys_Enable_IOTimerInterrupt_HDI (void) [inline]

Enables the Timer1 interrupt

Enables the Timer1 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 143 of file io_HDI.c.

8.24.4.4 void Sys_Force_IOTimerInterrupt_HDI (void) [inline]

forces the Timer1 interrupt

forces the Timer1 interrupt to occur.

Definition at line 152 of file io_HDI.c.

8.24.4.5 void Sys_Init_IOTimer_HDI ( ) [inline]

initialises the I/O Timer

This function initialises the I/O Timer.

Definition at line 35 of file io_HDI.c.

8.24.4.6 void Sys_IOTimer_code_HDI ( ) [inline]

execution of all I/O handlers.

This function is executed every time the I/O timer is active and executes all I/O handlers

Definition at line 162 of file io_HDI.c.

8.24.4.7 void Sys_Reset_IOTimer_HDI (void) [inline]

resets the I/O Timer

This function resets the I/O Timer.

Definition at line 99 of file io_HDI.c.

8.24.4.8 void Sys_Start_IOTimer_HDI ( ) [inline]

starts the I/O Timer

This function starts the I/O Timer.

Definition at line 60 of file io_HDI.c.

8.24.4.9 void Sys_Stop_IOTimer_HDI (void) [inline]

stops the I/O Timer

This function stops the I/O Timer.

Definition at line 73 of file io_HDI.c.

8.24.5 Variable Documentation

8.24.5.1 sys_periodical_IOHandler* sys_iohandlers

List of I/O handlers

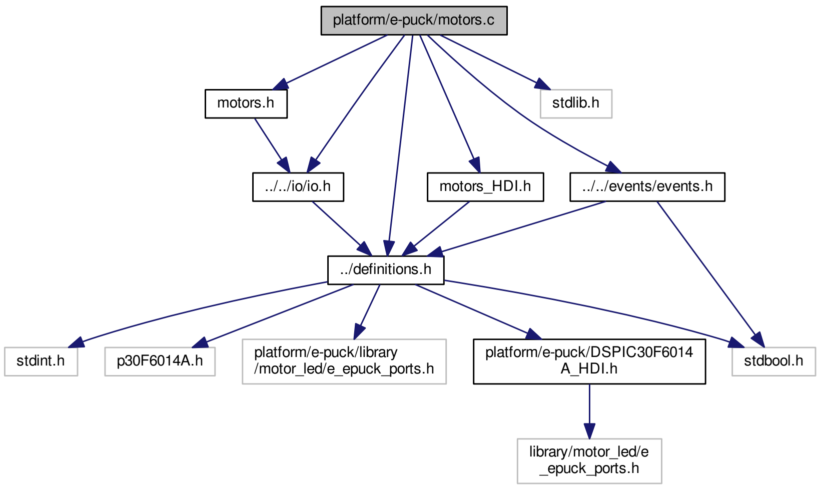
Definition at line 26 of file io_HDI.c.

8.25 platform/e-puck/motors.c File Reference

This file provides the function needed to actuate the motors.

```
#include "motors.h"
#include "motors_HDI.h"
#include "../../io/io.h"
#include "../../events/events.h"
#include "../../definitions.h"
#include <stdlib.h>
```

Include dependency graph for motors.c:



Data Structures

- struct sys_motors_s

  This struct contains speed for motors.

Macros

- #define MAX_WHEEL_SPEED 128
- #define POWER_SAVE_WAIT 15

Typedefs

- typedef struct sys_motors_s sys_motors

  This struct contains speed for motors.

Functions

- void Sys_LeftMotor_Controller (void)
- void Sys_RightMotor_Controller (void)
- bool Sys_LeftMotor_EventHandler (uint16, uint16, sys_event_data ∗)
- bool Sys_RightMotor_EventHandler (uint16, uint16, sys_event_data ∗)
- void Sys_Init_Motors ()
- void Sys_LeftMotor_Reset ()
- void Sys_RightMotor_Reset ()
- void Sys_Set_LeftWheelSpeed (sint16 speed)
- void Sys_Set_RightWheelSpeed (sint16 speed)
- sint16 Sys_Get_LeftWheelSpeed (void)
- sint16 Sys_Get_RightWheelSpeed (void)

8.25.1 Detailed Description

This file provides the function needed to actuate the motors.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org
Gabriel Kapellmann Zafra <gkapellmann@gmail.com >

Version

1.0

Date

30 July 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.25.2 Macro Definition Documentation

8.25.2.1 #define MAX_WHEEL_SPEED 128

Maximum wheel speed in steps

Definition at line 27 of file motors.c.

8.25.2.2 #define POWER_SAVE_WAIT 15

amount of steps needed to move the motor one step further

Definition at line 28 of file motors.c.

8.25.3 Typedef Documentation

8.25.3.1 typedef struct sys_motors_s sys_motors

This struct contains speed for motors.

8.25.4 Function Documentation

8.25.4.1 sint16 Sys_Get_LeftWheelSpeed (void)

returns the left wheel speed

This function returns the speed of the left motor.

Definition at line 281 of file motors.c.

8.25.4.2 sint16 Sys_Get_RightWheelSpeed (void)

returns the right wheel speed

This function returns the speed of the right motor.

Definition at line 291 of file motors.c.

8.25.4.3 void Sys_Init_Motors (void)

Initialises the Motor Module

This function initialises the motor module including both left and right motor.

Definition at line 52 of file motors.c.

8.25.4.4 void Sys_LeftMotor_Controller ( )

I/O handler for the left motor

This function controls the speed of the left motor. The speed is set by moving the to the next step within the appropriate time step.

Definition at line 116 of file motors.c.

8.25.4.5 bool Sys_LeftMotor_EventHandler (uint16 *pid*, uint16 *eventID*, sys_event_data ∗ *data*)

Left motor event handler to set the speed

This function sets the left motor speed that is received by the event SYS_EVENT_IO_MOTOR_LEFT.

Parameters

`in`	*pid*	the process id to which the event handler is registered
`in`	*eventID*	the event id which identifies the event that is handled
`in`	*data*	the event data that contain the motor speed.

Definition at line 215 of file motors.c.

8.25.4.6 void Sys_LeftMotor_Reset ( ) `[inline]`

resets the left motor

This function resets the left motor to a reset state.

Definition at line 96 of file motors.c.

8.25.4.7 void Sys_RightMotor_Controller ( )

I/O handler for the right motor

This function controls the speed of the right motor. The speed is set by moving the to the next step within the appropriate time step.

Definition at line 163 of file motors.c.

8.25.4.8 bool Sys_RightMotor_EventHandler (uint16 *pid*, uint16 *eventID*, sys_event_data ∗ *data*)

Right motor event handler to set the speed

This function sets the right motor speed that is received by the event SYS_EVENT_IO_MOTOR_RIGHT.

Parameters

`in`	*pid*	the process id to which the event handler is registered

in	*eventID*	the event id which identifies the event that is handled
in	*data*	the event data that contain the motor speed.

Definition at line 230 of file motors.c.

8.25.4.9 void Sys_RightMotor_Reset ( ) [inline]

resets the right motor

This function resets the right motor to a reset state.

Definition at line 106 of file motors.c.

8.25.4.10 void Sys_Set_LeftWheelSpeed (sint16 *speed*)

sets left wheel speed

This function sets the value for the speed of the left motor.

Parameters

speed	of the left wheel

Definition at line 246 of file motors.c.

8.25.4.11 void Sys_Set_RightWheelSpeed (sint16 *speed*)

sets right wheel speed

This function sets the value for the speed of the right motor.

Parameters

speed	of the right wheel

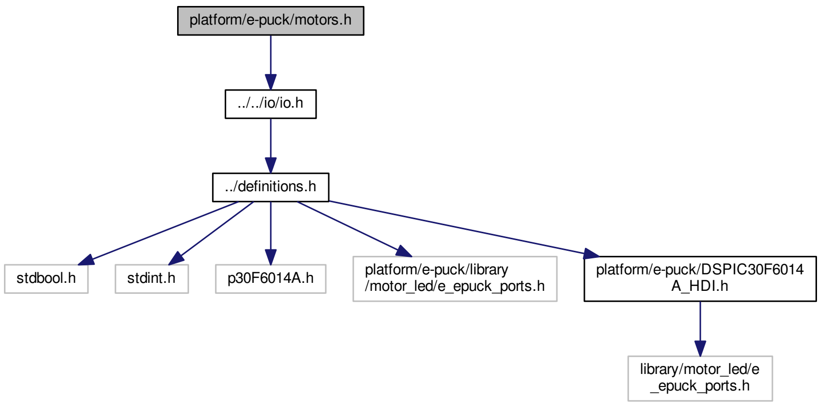
Definition at line 264 of file motors.c.

8.26 platform/e-puck/motors.h File Reference

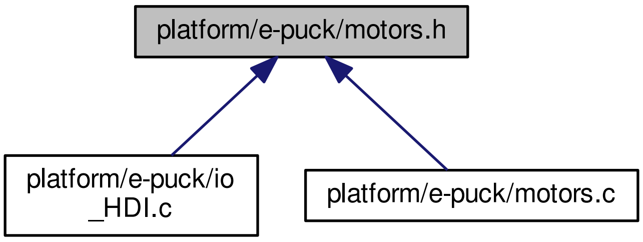
This file provides the function needed to actuate the motors.

```
#include "../../io/io.h"
```

Include dependency graph for motors.h:



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



Macros

- #define MAX_WHEEL_SPEED_MM_S 129 /∗mm/s∗/

Functions

- void Sys_Init_Motors (void)
- void Sys_Set_LeftWheelSpeed (sint16 speed)
- void Sys_Set_RightWheelSpeed (sint16 speed)
- sint16 Sys_Get_LeftWheelSpeed (void)
- sint16 Sys_Get_RightWheelSpeed (void)

8.26.1 Detailed Description

This file provides the function needed to actuate the motors.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org
Gabriel Kapellmann Zafra <gkapellmann@gmail.com>

Version

1.0

Date

30 July 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.26.2 Macro Definition Documentation

8.26.2.1 #define MAX_WHEEL_SPEED_MM_S 129 /*mm/s*/

Maximum wheel speed in mm/s

Definition at line 46 of file motors.h.

8.26.3 Function Documentation

8.26.3.1 sint16 Sys_Get_LeftWheelSpeed (void)

returns the left wheel speed

This function returns the speed of the left motor.

Definition at line 281 of file motors.c.

8.26.3.2 sint16 Sys_Get_RightWheelSpeed (void)

returns the right wheel speed

This function returns the speed of the right motor.

Definition at line 291 of file motors.c.

8.26.3.3 void Sys_Init_Motors (void)

Initialises the Motor Module

This function initialises the motor module including both left and right motor.

Definition at line 52 of file motors.c.

8.26.3.4 void Sys_Set_LeftWheelSpeed (sint16 *speed*)

sets left wheel speed

This function sets the value for the speed of the left motor.

Parameters

speed	of the left wheel

Definition at line 246 of file motors.c.

8.26.3.5 void Sys_Set_RightWheelSpeed (sint16 *speed*)

sets right wheel speed

This function sets the value for the speed of the right motor.

Parameters

speed	of the right wheel

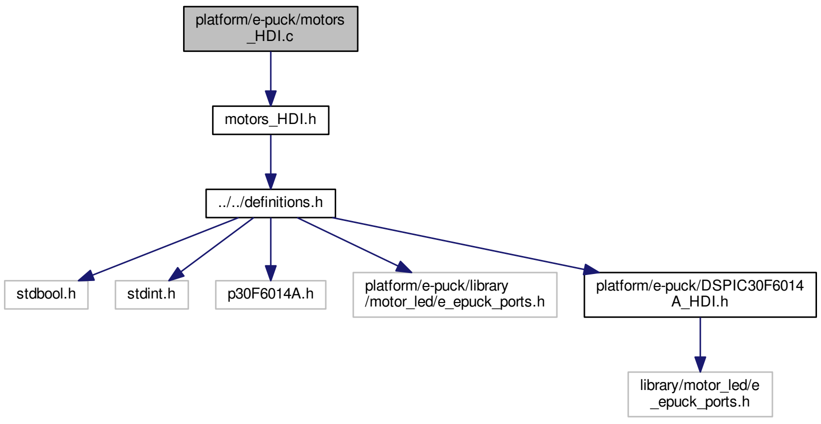
Definition at line 264 of file motors.c.

8.27 platform/e-puck/motors_HDI.c File Reference

Hardware dependent implementations to actuate the motors.

```
#include "motors_HDI.h"
```

Include dependency graph for motors_HDI.c:


platform/e-puck/motors _HDI.c
motors_HDI.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

Functions

- void Sys_LeftMotor_SetPhase_HDI (sint8 phase)
- void Sys_RightMotor_SetPhase_HDI (sint8 phase)

8.27.1 Detailed Description

Hardware dependent implementations to actuate the motors.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.27.2 Function Documentation

8.27.2.1 void Sys_LeftMotor_SetPhase_HDI (sint8 *phase*) [inline]

sets the left motor phase

This function sets the left motor phase

Parameters

in	*phase*	indicates the phase of the left motor

Definition at line 28 of file motors_HDI.c.

8.27.2.2 void Sys_RightMotor_SetPhase_HDI (sint8 *phase*) [inline]

sets the right motor phase

This function sets the right motor phase

Parameters

in	*phase*	indicates the phase of the right motor

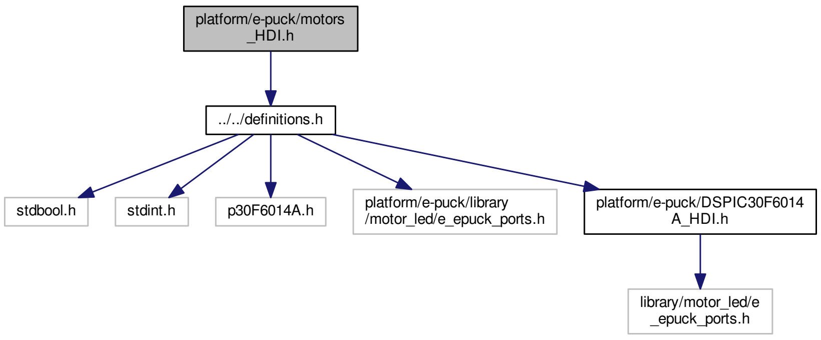
Definition at line 82 of file motors_HDI.c.

8.28 platform/e-puck/motors_HDI.h File Reference

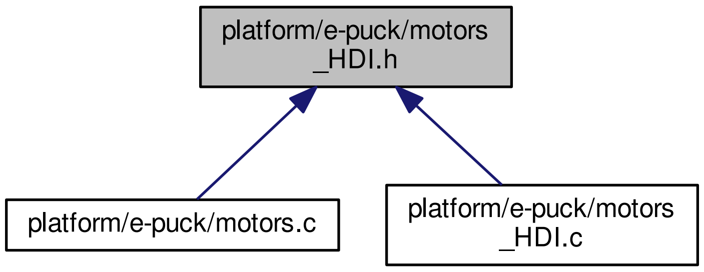
Hardware dependent implementations to actuate the motors.

```
#include "../../definitions.h"
```

Include dependency graph for motors_HDI.h:


platform/e-puck/motors_HDI.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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



Macros

- #define MOTORPHASE_RESET -1

Functions

- void Sys_LeftMotor_SetPhase_HDI (sint8 phase)
- void Sys_RightMotor_SetPhase_HDI (sint8 phase)

8.28.1 Detailed Description

Hardware dependent implementations to actuate the motors.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.28.2 Macro Definition Documentation

8.28.2.1 #define MOTORPHASE_RESET -1

the reset value for the motor phase

Definition at line 26 of file motors_HDI.h.

8.28.3 Function Documentation

8.28.3.1 void Sys_LeftMotor_SetPhase_HDI (sint8 *phase*) [inline]

sets the left motor phase

This function sets the left motor phase

Parameters

in	*phase*	indicates the phase of the left motor

Definition at line 28 of file motors_HDI.c.

8.28.3.2 void Sys_RightMotor_SetPhase_HDI (sint8 *phase*) [inline]

sets the right motor phase

This function sets the right motor phase

Parameters

in	*phase*	indicates the phase of the right motor

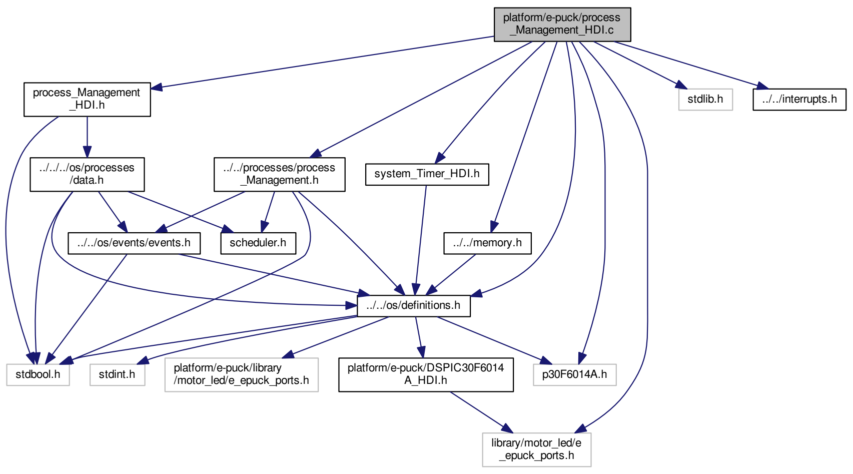
Definition at line 82 of file motors_HDI.c.

8.29 platform/e-puck/process_Management_HDI.c File Reference

Hardware dependent implementations to manage processes (e.g. task swichting)

```
#include "process_Management_HDI.h"
#include "../../processes/process_Management.h"
#include <stdlib.h>
#include "system_Timer_HDI.h"
#include "../../interrupts.h"
#include "../../memory.h"
#include "../../definitions.h"
#include <p30F6014A.h>
#include "library/motor_led/e_epuck_ports.h"
```

Include dependency graph for process_Management_HDI.c:


platform/e-puck/process_Management_HDI.c
process_Management_HDI.h
stdlib.h
../../interrupts.h
../../../os/processes/data.h
../../processes/process_Management.h
system_Timer_HDI.h
../../os/events/events.h
scheduler.h
../../memory.h
../../os/definitions.h
stdbool.h
stdint.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
p30F6014A.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_Init_Process_Management_HDI ()
- bool Sys_Start_Process_HDI (pFunction function)
- void Sys_Save_Running_Process_HDI ()
- void Sys_Change_Stack_HDI (unsigned short fp, unsigned short sp, unsigned short lm)
- void Sys_Switch_Process_HDI (sys_pcb_list_element ∗new_process)

8.29.1 Detailed Description

Hardware dependent implementations to manage processes (e.g. task swichting)

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

{08 July 2014}

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.29.2 Function Documentation

8.29.2.1 void Sys_Change_Stack_HDI (unsigned short *fp*, unsigned short *sp*, unsigned short *lm*)

This function changes stackpointers to the new stack

This function changes stackpointers to the new stack

Parameters

in	*fp*	FramePointer address
in	*sp*	StackPointer address
in	*lm*	StackPointer Limit

Definition at line 215 of file process_Management_HDI.c.

8.29.2.2 void Sys_Init_Process_Management_HDI (void)

This function initialises the process management

This function initialises the process management and creates the first elements in the linked list

Definition at line 44 of file process_Management_HDI.c.

8.29.2.3 void Sys_Save_Running_Process_HDI (void) [inline]

This function stores all registers and information of the running process into the corresponding struct

This function stores all registers and information of the running process into the corresponding struct

Definition at line 151 of file process_Management_HDI.c.

8.29.2.4 bool Sys_Start_Process_HDI (pFunction *function*)

This function creates a new sys_process_control_block and add all needed info

This function creates a new sys_process_control_block (in a sys_process_control_block_list_element) which contains all information wich is used to execute this process.

Parameters

in	*function*	This argument points to a function in memory which should be executed as an new task

Definition at line 95 of file process_Management_HDI.c.

8.29.2.5 void Sys_Switch_Process_HDI (sys_pcb_list_element ∗ *new_process*)

This function switches from sys_running_process to new_process

This function switches from sys_running_process to new_process

Parameters

in	*new_process*	pointer to the process which should be executed

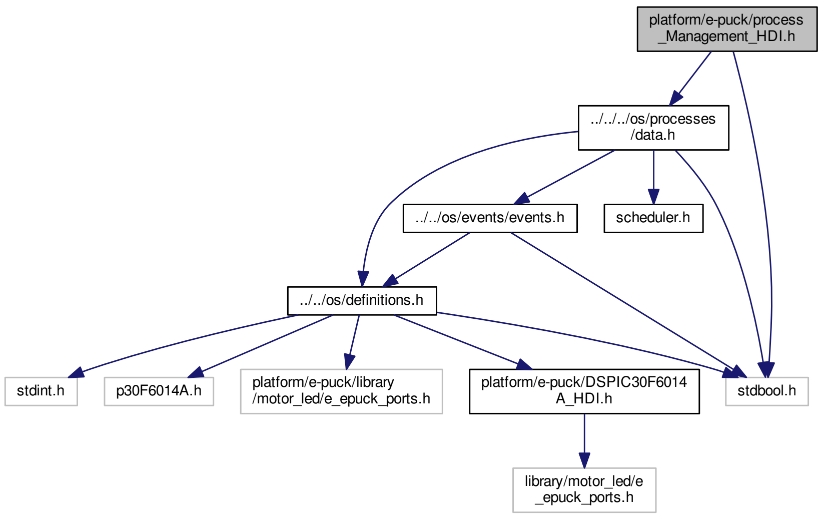
Definition at line 248 of file process_Management_HDI.c.

8.30 platform/e-puck/process_Management_HDI.h File Reference

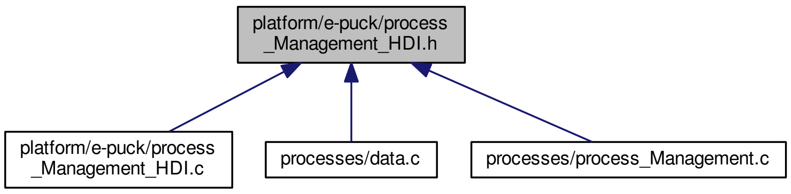
Hardware dependent implementations to manage processes (e.g. task swichting)

```
#include "../../../os/processes/data.h"
#include <stdbool.h>
```

Include dependency graph for process_Management_HDI.h:


platform/e-puck/process_Management_HDI.h
../../../os/processes/data.h
../../os/events/events.h
scheduler.h
../../os/definitions.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
stdbool.h
library/motor_led/e_epuck_ports.h

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


platform/e-puck/process_Management_HDI.h
platform/e-puck/process_Management_HDI.c
processes/data.c
processes/process_Management.c

Functions

- void Sys_Init_Process_Management_HDI (void)
- bool Sys_Start_Process_HDI (pFunction function)
- void Sys_Save_Running_Process_HDI (void)
- void Sys_Change_Stack_HDI (unsigned short fp, unsigned short sp, unsigned short lm)
- void Sys_Switch_Process_HDI (sys_pcb_list_element ∗new_process)

8.30.1 Detailed Description

Hardware dependent implementations to manage processes (e.g. task swichting)

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

{08 July 2014}

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.30.2 Function Documentation

8.30.2.1 void Sys_Change_Stack_HDI (unsigned short *fp*, unsigned short *sp*, unsigned short *lm*)

This function changes stackpointers to the new stack

This function changes stackpointers to the new stack

Parameters

in	*fp*	FramePointer address
in	*sp*	StackPointer address
in	*lm*	StackPointer Limit

Definition at line 215 of file process_Management_HDI.c.

8.30.2.2 void Sys_Init_Process_Management_HDI (void)

This function initialises the process management

This function initialises the process management and creates the first elements in the linked list

Definition at line 44 of file process_Management_HDI.c.

8.30.2.3 void Sys_Save_Running_Process_HDI (void) `[inline]`

This function stores all registers and information of the running process into the corresponding struct

This function stores all registers and information of the running process into the corresponding struct

Definition at line 151 of file process_Management_HDI.c.

8.30.2.4 bool Sys_Start_Process_HDI (pFunction *function*)

This function creates a new sys_process_control_block and add all needed info

This function creates a new sys_process_control_block (in a sys_process_control_block_list_element) which contains all information wich is used to execute this process.

Parameters

in	*function*	This argument points to a function in memory which should be executed as an new task

Definition at line 95 of file process_Management_HDI.c.

8.30.2.5 void Sys_Switch_Process_HDI (sys_pcb_list_element * *new_process*)

This function switches from sys_running_process to new_process

This function switches from sys_running_process to new_process

Parameters

in	*new_process*	pointer to the process which should be executed

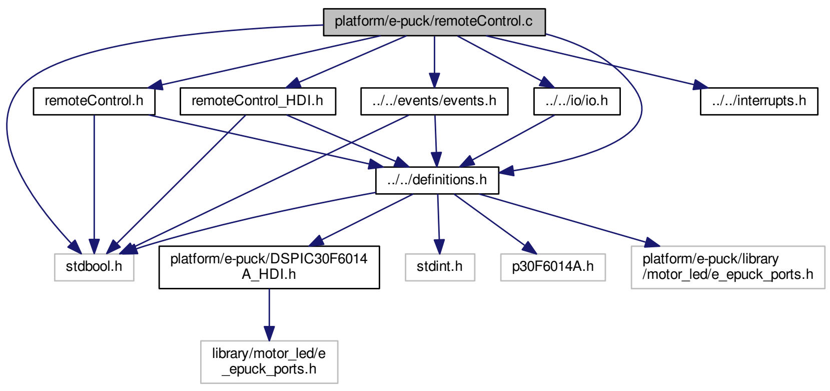
Definition at line 248 of file process_Management_HDI.c.

8.31 platform/e-puck/remoteControl.c File Reference

This file includes functions needed to receive and decode messages from a remote control.

```
#include "remoteControl.h"
#include "remoteControl_HDI.h"
#include <stdbool.h>
#include "../../io/io.h"
#include "../../events/events.h"
#include "../../definitions.h"
#include "../../interrupts.h"
```

Include dependency graph for remoteControl.c:


platform/e-puck/remoteControl.c
remoteControl.h
remoteControl_HDI.h
../../events/events.h
../../io/io.h
../../interrupts.h
../../definitions.h
stdbool.h
platform/e-puck/DSPIC30F6014A_HDI.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_Init_RemoteControl (void)
- void Sys_Start_RemoteControl (void)
- void Sys_Receive_RemoteControl_Msg ()
- bool Sys_HasRemoteC_Sent_New_Data ()
- uint8 Sys_RemoteC_Get_CheckBit ()
- uint8 Sys_RemoteC_Get_Address ()
- uint8 Sys_RemoteC_Get_Data ()

8.31.1 Detailed Description

This file includes functions needed to receive and decode messages from a remote control.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org
Yuri Kaszubowski Lopes yurikazuba@gmail.com

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.31.2 Function Documentation

8.31.2.1 bool Sys_HasRemoteC_Sent_New_Data ( )

returns if a new command was read

This function returns true if a new remote control command has arrived

Returns

bool true if a new remote control command has arrived

Definition at line 124 of file remoteControl.c.

8.31.2.2 void Sys_Init_RemoteControl (void) [inline]

Initialises the remote control handler

This function initialises the handler of the remote control to receive signals from the remote control.

Definition at line 37 of file remoteControl.c.

8.31.2.3 void Sys_Receive_RemoteControl_Msg (void)

handles incoming remote control signals

This function reads a remote control signal and reads it's transmitted value. When a signal arrives, an external interrupt is triggered. The remaining values are obtained by using time not interrupts.

Definition at line 57 of file remoteControl.c.

8.31.2.4 uint8 Sys_RemoteC_Get_Address (void)

returns the address of the remote control

This function returns the address of an remote control.

Returns

address of the remote control

Definition at line 146 of file remoteControl.c.

8.31.2.5 uint8 Sys_RemoteC_Get_CheckBit (void)

returns the check bit value

This function returns the check bit value of an remote control to indicate if a button was pressed continuously or sequential.

Returns

bit to indicate the check bit

Definition at line 135 of file remoteControl.c.

8.31.2.6 uint8 Sys_RemoteC_Get_Data (void)

returns the value received by the remote control

returns the value received by the remote control

Returns

value received by the remote control

Definition at line 157 of file remoteControl.c.

8.31.2.7 void Sys_Start_RemoteControl (void) [inline]

start the remote control handler

This function start the handler of the remote control to receive signals from the remote control.

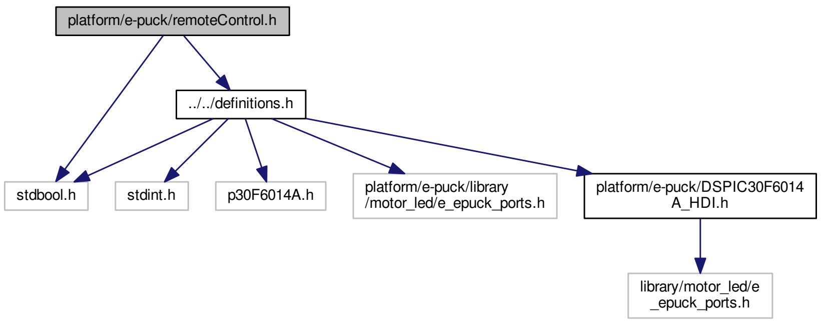
Definition at line 47 of file remoteControl.c.

8.32 platform/e-puck/remoteControl.h File Reference

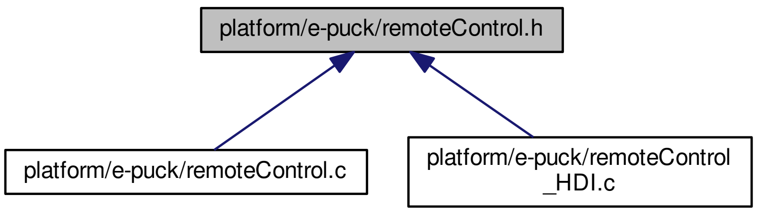
This file includes functions needed to receive and decode messages from a remote control.

```
#include <stdbool.h>
#include "../../definitions.h"
```

Include dependency graph for remoteControl.h:


platform/e-puck/remoteControl.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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


platform/e-puck/remoteControl.h
platform/e-puck/remoteControl.c
platform/e-puck/remoteControl_HDI.c

Macros

- #define RC_BUTTON_STANDBY 12
- #define RC_BUTTON_SCREEN 11
- #define RC_BUTTON_LANG 15
- #define RC_BUTTON_SUBTTL 31
- #define RC_BUTTON_INTERNET 46
- #define RC_BUTTON_RED 55
- #define RC_BUTTON_GREEN 54

- #define RC_BUTTON_YELLOW 50
- #define RC_BUTTON_BLUE 52
- #define RC_BUTTON_0 0
- #define RC_BUTTON_1 1
- #define RC_BUTTON_2 2
- #define RC_BUTTON_3 3
- #define RC_BUTTON_4 4
- #define RC_BUTTON_5 5
- #define RC_BUTTON_6 6
- #define RC_BUTTON_7 7
- #define RC_BUTTON_8 8
- #define RC_BUTTON_9 9
- #define RC_BUTTON_TELE_TEXT 60
- #define RC_BUTTON_SWAP 34
- #define RC_BUTTON_OK 53
- #define RC_BUTTON_CURSOR_UP 20
- #define RC_BUTTON_CURSOR_DOWN 19
- #define RC_BUTTON_CURSOR_LEFT 21
- #define RC_BUTTON_CURSOR_RIGHT 22
- #define RC_BUTTON_BACK 10
- #define RC_BUTTON_MENU 48
- #define RC_BUTTON_EPG 47
- #define RC_BUTTON_FAV 40
- #define RC_BUTTON_SOURCE 56
- #define RC_BUTTON_INFO 18
- #define RC_BUTTON_PRESETS 14
- #define RC_BUTTON_SLEEP 42
- #define RC_BUTTON_VOLUME_UP 16
- #define RC_BUTTON_VOLUME_DOWN 17
- #define RC_BUTTON_MUTE 13
- #define RC_BUTTON_CHANNEL_UP 32
- #define RC_BUTTON_CHANNEL_DOWN 33
- #define RC_BUTTON_PAUSE 48
- #define RC_BUTTON_REWIND 50
- #define RC_BUTTON_WIND 52
- #define RC_BUTTON_PLAY 53
- #define RC_BUTTON_STOP 54
- #define RC_BUTTON_RECORD 55

Functions

- void Sys_Init_RemoteControl (void)
- void Sys_Start_RemoteControl (void)
- bool Sys_RemoteC_Received_New_Data (void)
- uint8 Sys_RemoteC_Get_CheckBit (void)
- uint8 Sys_RemoteC_Get_Address (void)
- uint8 Sys_RemoteC_Get_Data (void)
- void Sys_Receive_RemoteControl_Msg (void)

8.32.1 Detailed Description

This file includes functions needed to receive and decode messages from a remote control.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org
Yuri Kaszubowski Lopes yurikazuba@gmail.com

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.32.2 Macro Definition Documentation

8.32.2.1 #define RC_BUTTON_0 0

Value for the 0 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 61 of file remoteControl.h.

8.32.2.2 #define RC_BUTTON_1 1

Value for the 1 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 62 of file remoteControl.h.

8.32.2.3 #define RC_BUTTON_2 2

Value for the 2 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 63 of file remoteControl.h.

8.32.2.4 #define RC_BUTTON_3 3

Value for the 3 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 64 of file remoteControl.h.

8.32.2.5 #define RC_BUTTON_4 4

Value for the 4 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 65 of file remoteControl.h.

8.32.2.6 #define RC_BUTTON_5 5

Value for the 5 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 66 of file remoteControl.h.

8.32.2.7 #define RC_BUTTON_6 6

Value for the 6 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 67 of file remoteControl.h.

8.32.2.8 #define RC_BUTTON_7 7

Value for the 7 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 68 of file remoteControl.h.

8.32.2.9 #define RC_BUTTON_8 8

Value for the 8 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 69 of file remoteControl.h.

8.32.2.10 #define RC_BUTTON_9 9

Value for the 9 button (RC-5 coding for a Toshiba RC-3910)

Definition at line 70 of file remoteControl.h.

8.32.2.11 #define RC_BUTTON_BACK 10

Value for the back button (RC-5 coding for a Toshiba RC-3910)

Definition at line 79 of file remoteControl.h.

8.32.2.12 #define RC_BUTTON_BLUE 52

Value for the blue button (RC-5 coding for a Toshiba RC-3910)

Definition at line 59 of file remoteControl.h.

8.32.2.13 #define RC_BUTTON_CHANNEL_DOWN 33

Value for the channel down button (RC-5 coding for a Toshiba RC-3910)

Definition at line 93 of file remoteControl.h.

8.32.2.14 #define RC_BUTTON_CHANNEL_UP 32

Value for the channel up button (RC-5 coding for a Toshiba RC-3910)

Definition at line 92 of file remoteControl.h.

8.32.2.15 #define RC_BUTTON_CURSOR_DOWN 19

Value for the courser down button (RC-5 coding for a Toshiba RC-3910)

Definition at line 76 of file remoteControl.h.

8.32.2.16 #define RC_BUTTON_CURSOR_LEFT 21

Value for the courser left button (RC-5 coding for a Toshiba RC-3910)

Definition at line 77 of file remoteControl.h.

8.32.2.17 #define RC_BUTTON_CURSOR_RIGHT 22

Value for the courser right button (RC-5 coding for a Toshiba RC-3910)

Definition at line 78 of file remoteControl.h.

8.32.2.18 #define RC_BUTTON_CURSOR_UP 20

Value for the coursor up button (RC-5 coding for a Toshiba RC-3910)

Definition at line 75 of file remoteControl.h.

8.32.2.19 #define RC_BUTTON_EPG 47

Value for the epg button (RC-5 coding for a Toshiba RC-3910)

Definition at line 81 of file remoteControl.h.

8.32.2.20 #define RC_BUTTON_FAV 40

Value for the favourite button (RC-5 coding for a Toshiba RC-3910)

Definition at line 82 of file remoteControl.h.

8.32.2.21 #define RC_BUTTON_GREEN 54

Value for the green button (RC-5 coding for a Toshiba RC-3910)

Definition at line 57 of file remoteControl.h.

8.32.2.22 #define RC_BUTTON_INFO 18

Value for the info button (RC-5 coding for a Toshiba RC-3910)

Definition at line 85 of file remoteControl.h.

8.32.2.23 #define RC_BUTTON_INTERNET 46

Value for the internet button (RC-5 coding for a Toshiba RC-3910)

Definition at line 54 of file remoteControl.h.

8.32.2.24 #define RC_BUTTON_LANG 15

Value for the language button (RC-5 coding for a Toshiba RC-3910)

Definition at line 52 of file remoteControl.h.

8.32.2.25 #define RC_BUTTON_MENU 48

Value for the menu button (RC-5 coding for a Toshiba RC-3910)

Definition at line 80 of file remoteControl.h.

8.32.2.26 #define RC_BUTTON_MUTE 13

Value for the mute button (RC-5 coding for a Toshiba RC-3910)

Definition at line 91 of file remoteControl.h.

8.32.2.27 #define RC_BUTTON_OK 53

Value for the OK button (RC-5 coding for a Toshiba RC-3910)

Definition at line 74 of file remoteControl.h.

8.32.2.28 #define RC_BUTTON_PAUSE 48

Value for the pause button (RC-5 coding for a Toshiba RC-3910)

Definition at line 96 of file remoteControl.h.

8.32.2.29 #define RC_BUTTON_PLAY 53

Value for the play button (RC-5 coding for a Toshiba RC-3910)

Definition at line 99 of file remoteControl.h.

8.32.2.30 #define RC_BUTTON_PRESETS 14

Value for the preset button (RC-5 coding for a Toshiba RC-3910)

Definition at line 86 of file remoteControl.h.

8.32.2.31 #define RC_BUTTON_RECORD 55

Value for the record button (RC-5 coding for a Toshiba RC-3910)

Definition at line 101 of file remoteControl.h.

8.32.2.32 #define RC_BUTTON_RED 55

Value for the red button (RC-5 coding for a Toshiba RC-3910)

Definition at line 56 of file remoteControl.h.

8.32.2.33 #define RC_BUTTON_REWIND 50

Value for the rewind button (RC-5 coding for a Toshiba RC-3910)

Definition at line 97 of file remoteControl.h.

8.32.2.34 #define RC_BUTTON_SCREEN 11

Value for the screen button (RC-5 coding for a Toshiba RC-3910)

Definition at line 51 of file remoteControl.h.

8.32.2.35 #define RC_BUTTON_SLEEP 42

Value for the sleep button (RC-5 coding for a Toshiba RC-3910)

Definition at line 87 of file remoteControl.h.

8.32.2.36 #define RC_BUTTON_SOURCE 56

Value for the source button (RC-5 coding for a Toshiba RC-3910)

Definition at line 84 of file remoteControl.h.

8.32.2.37 #define RC_BUTTON_STANDBY 12

Value for the standby button (RC-5 coding for a Toshiba RC-3910)

Definition at line 49 of file remoteControl.h.

8.32.2.38 #define RC_BUTTON_STOP 54

Value for the stop button (RC-5 coding for a Toshiba RC-3910)

Definition at line 100 of file remoteControl.h.

8.32.2.39 #define RC_BUTTON_SUBTTL 31

Value for the subtitle button (RC-5 coding for a Toshiba RC-3910)

Definition at line 53 of file remoteControl.h.

8.32.2.40 #define RC_BUTTON_SWAP 34

Value for the swap button (RC-5 coding for a Toshiba RC-3910)

Definition at line 72 of file remoteControl.h.

8.32.2.41 #define RC_BUTTON_TELE_TEXT 60

Value for the tele text button (RC-5 coding for a Toshiba RC-3910)

Definition at line 71 of file remoteControl.h.

8.32.2.42 #define RC_BUTTON_VOLUME_DOWN 17

Value for the volume down button (RC-5 coding for a Toshiba RC-3910)

Definition at line 90 of file remoteControl.h.

8.32.2.43 #define RC_BUTTON_VOLUME_UP 16

Value for the volume up button (RC-5 coding for a Toshiba RC-3910)

Definition at line 89 of file remoteControl.h.

8.32.2.44 #define RC_BUTTON_WIND 52

Value for the wind button (RC-5 coding for a Toshiba RC-3910)

Definition at line 98 of file remoteControl.h.

8.32.2.45 #define RC_BUTTON_YELLOW 50

Value for the yellow button (RC-5 coding for a Toshiba RC-3910)

Definition at line 58 of file remoteControl.h.

8.32.3 Function Documentation

8.32.3.1 void Sys_Init_RemoteControl (void) [inline]

Initialises the remote control handler

This function initialises the handler of the remote control to receive signals from the remote control.

Definition at line 37 of file remoteControl.c.

8.32.3.2 void Sys_Receive_RemoteControl_Msg (void)

handles incoming remote control signals

This function reads a remote control signal and reads it's transmitted value. When a signal arrives, an external interrupt is triggered. The remaining values are obtained by using time not interrupts.

Definition at line 57 of file remoteControl.c.

8.32.3.3 uint8 Sys_RemoteC_Get_Address (void)

returns the address of the remote control

This function returns the address of an remote control.

Returns

address of the remote control

Definition at line 146 of file remoteControl.c.

8.32.3.4 uint8 Sys_RemoteC_Get_CheckBit (void)

returns the check bit value

This function returns the check bit value of an remote control to indicate if a button was pressed continuously or sequential.

Returns

bit to indicate the check bit

Definition at line 135 of file remoteControl.c.

8.32.3.5 uint8 Sys_RemoteC_Get_Data (void)

returns the value received by the remote control

returns the value received by the remote control

Returns

value received by the remote control

Definition at line 157 of file remoteControl.c.

8.32.3.6 bool Sys_RemoteC_Received_New_Data (void)

8.32.3.7 void Sys_Start_RemoteControl (void) [inline]

start the remote control handler

This function start the handler of the remote control to receive signals from the remote control.

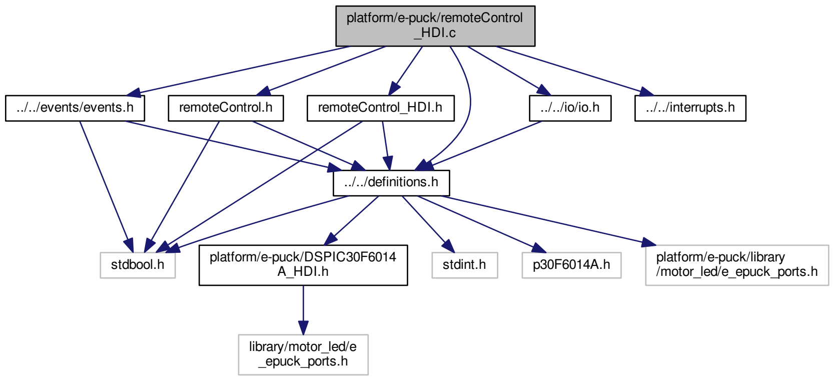
Definition at line 47 of file remoteControl.c.

8.33 platform/e-puck/remoteControl_HDI.c File Reference

Hardware dependent implementations to receive and decode messages from a remote control.

```
#include "remoteControl_HDI.h"
#include "../../definitions.h"
#include "../../interrupts.h"
#include "../../io/io.h"
#include "remoteControl.h"
#include "../../events/events.h"
```

Include dependency graph for remoteControl_HDI.c:


platform/e-puck/remoteControl_HDI.c
../../events/events.h
remoteControl.h
remoteControl_HDI.h
../../io/io.h
../../interrupts.h
../../definitions.h
stdbool.h
platform/e-puck/DSPIC30F6014A_HDI.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_Init_RemoteControl_HDI (void)
- void Sys_Start_RemoteControl_HDI (void)
- void __attribute__ ((__interrupt__, auto_psv))

Variables

- bool message_arriving = false
- sint8 waiting_cycles = 20
- uint16 rx_buffer = 0
- bool isNewDataAvailable = false
- sint8 receiving_bit = RC_NOT_STARTED

8.33.1 Detailed Description

Hardware dependent implementations to receive and decode messages from a remote control.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`
Yuri Kaszubowski Lopes `yurikazuba@gmail.com`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.33.2 Function Documentation

8.33.2.1 void __attribute__ ((__interrupt__, auto_psv))

external interrupt handler for the remote control

This function is executed at the arrival of a new remote control message.

Definition at line 74 of file remoteControl_HDI.c.

8.33.2.2 void Sys_Init_RemoteControl_HDI (void) `[inline]`

Initialises the remote control handler

This function initialises the handler of the remote control to receive signals from the remote control.

Definition at line 43 of file remoteControl_HDI.c.

8.33.2.3 void Sys_Start_RemoteControl_HDI (void) [inline]

start the remote control handler

This function start the handler of the remote control to receive signals from the remote control. clear to IRQ flag

Definition at line 63 of file remoteControl_HDI.c.

8.33.3 Variable Documentation

8.33.3.1 bool isNewDataAvailable = false

a flag to indicate that a new message was received

Definition at line 33 of file remoteControl_HDI.c.

8.33.3.2 bool message_arriving = false

A flag that is set as soon as a messgage is recieved

Definition at line 28 of file remoteControl_HDI.c.

8.33.3.3 sint8 receiving_bit = RC_NOT_STARTED

State indicator (for the state machine)

Definition at line 35 of file remoteControl_HDI.c.

8.33.3.4 uint16 rx_buffer = 0

The initial state of the state machine to decode a remote control message

Definition at line 31 of file remoteControl_HDI.c.

8.33.3.5 sint8 waiting_cycles = 20

The cycles that need to be waited until the next stage (set for 100us)

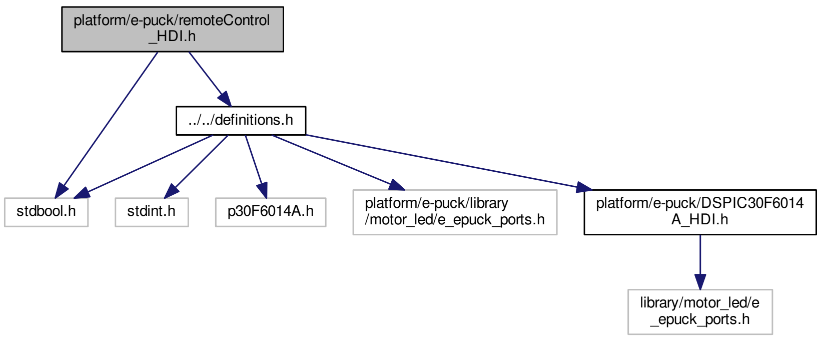
Definition at line 29 of file remoteControl_HDI.c.

8.34 platform/e-puck/remoteControl_HDI.h File Reference

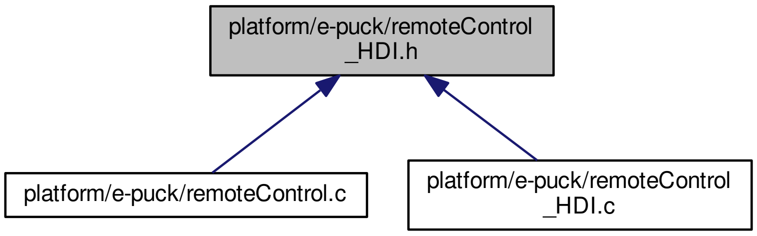
Hardware dependent implementations to receive and decode messages from a remote control.

```
#include <stdbool.h>
#include "../../definitions.h"
```

Include dependency graph for remoteControl_HDI.h:


platform/e-puck/remoteControl_HDI.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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


platform/e-puck/remoteControl_HDI.h
platform/e-puck/remoteControl.c
platform/e-puck/remoteControl_HDI.c

Macros

- #define RC_WAIT_FOR_QUARTERBIT 4
- #define RC_WAIT_FOR_HALFBIT 8
- #define RC_WAIT_FOR_BIT 18
- #define RC_WAIT_INITIALLY RC_WAIT_FOR_BIT+RC_WAIT_FOR_QUARTERBIT
- #define RC_NOT_STARTED -1

Functions

- void Sys_Init_RemoteControl_HDI (void)
- void Sys_Start_RemoteControl_HDI (void)

Variables

- bool message_arriving
- sint8 waiting_cycles
- uint16 rx_buffer
- bool isNewDataAvailable
- sint8 receiving_bit

8.34.1 Detailed Description

Hardware dependent implementations to receive and decode messages from a remote control.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org
Yuri Kaszubowski Lopes yurikazuba@gmail.com

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.34.2 Macro Definition Documentation

8.34.2.1 #define RC_NOT_STARTED -1

The initial state of the state machine to decode a remote control message

Definition at line 35 of file remoteControl_HDI.h.

8.34.2.2 #define RC_WAIT_FOR_BIT 18

Cycles that are needed to wait a single bit duration

Definition at line 33 of file remoteControl_HDI.h.

8.34.2.3 #define RC_WAIT_FOR_HALFBIT 8

Cycles that are needed to wait a half of a single bit duration

Definition at line 32 of file remoteControl_HDI.h.

8.34.2.4 #define RC_WAIT_FOR_QUARTERBIT 4

Cycles that are needed to wait a quarter of a single bit duration

Definition at line 31 of file remoteControl_HDI.h.

8.34.2.5 #define RC_WAIT_INITIALLY RC_WAIT_FOR_BIT+RC_WAIT_FOR_QUARTERBIT

Cycles that are needed to wait at the beginning of a message

Definition at line 34 of file remoteControl_HDI.h.

8.34.3 Function Documentation

8.34.3.1 void Sys_Init_RemoteControl_HDI (void) [inline]

Initialises the remote control handler

This function initialises the handler of the remote control to receive signals from the remote control.

Definition at line 43 of file remoteControl_HDI.c.

8.34.3.2 void Sys_Start_RemoteControl_HDI (void) [inline]

start the remote control handler

This function start the handler of the remote control to receive signals from the remote control. clear to IRQ flag

Definition at line 63 of file remoteControl_HDI.c.

8.34.4 Variable Documentation

8.34.4.1 bool isNewDataAvailable

a flag to indicate that a new message was received

Definition at line 33 of file remoteControl_HDI.c.

8.34.4.2 bool message_arriving

A flag that is set as soon as a messgage is recieved

Definition at line 28 of file remoteControl_HDI.c.

8.34.4.3 sint8 receiving_bit

State indicator (for the state machine)

Definition at line 35 of file remoteControl_HDI.c.

8.34.4.4 uint16 rx_buffer

The initial state of the state machine to decode a remote control message

Definition at line 31 of file remoteControl_HDI.c.

8.34.4.5 sint8 waiting_cycles

The cycles that need to be waited until the next stage (set for 100us)

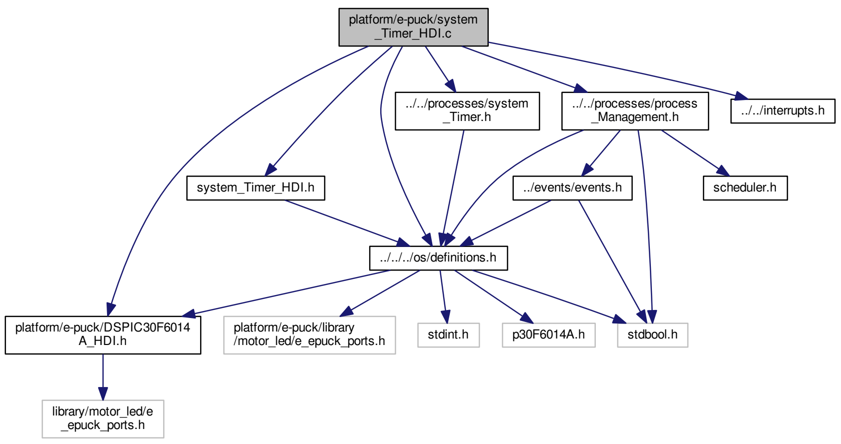
Definition at line 29 of file remoteControl_HDI.c.

8.35 platform/e-puck/system_Timer_HDI.c File Reference

Hardware dependent implementations to initialise, configure and the operating system.

```
#include "system_Timer_HDI.h"
#include "../../processes/system_Timer.h"
#include "../../processes/process_Management.h"
#include "DSPIC30F6014A_HDI.h"
#include "../../interrupts.h"
#include "../../definitions.h"
```

Include dependency graph for system_Timer_HDI.c:


platform/e-puck/system_Timer_HDI.c
../../processes/system_Timer.h
../../processes/process_Management.h
../../interrupts.h
system_Timer_HDI.h
../events/events.h
scheduler.h
../../../os/definitions.h
platform/e-puck/DSPIC30F6014A_HDI.h
platform/e-puck/library/motor_led/e_epuck_ports.h
stdint.h
p30F6014A.h
stdbool.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_Init_SystemTimer_HDI (pFunction scheduler)
- void Sys_Start_SystemTimer_HDI ()
- void Sys_Stop_SystemTimer_HDI ()
- void Sys_Continue_SystemTimer_HDI ()
- void Sys_Reset_SystemTimer_HDI ()
- void __attribute__ ((interrupt, no_auto_psv))
- void Sys_Disable_TimerInterrupt_HDI (void)
- void Sys_Enable_TimerInterrupt_HDI (void)
- void Sys_Force_TimerInterrupt_HDI (void)

Variables

- pFunction sys_process_scheduler = 0

8.35.1 Detailed Description

Hardware dependent implementations to initialise, configure and the operating system.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

07 July 2014

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.35.2 Function Documentation

8.35.2.1 void __attribute__ ((interrupt, no_auto_psv))

Interrupt Service Routine for the Timer2 HDI

This Function starts the task-scheduling algorithm

Interrupt Service Routine for the Timer2 HDI (alternate)

This Function starts the task-scheduling algorithm

Definition at line 112 of file system_Timer_HDI.c.

8.35.2.2 void Sys_Continue_SystemTimer_HDI ( ) `[inline]`

Deactivates the Timer2 Interrupt

This Function deactivated the Timer2 Interrupt

Definition at line 89 of file system_Timer_HDI.c.

8.35.2.3 void Sys_Disable_TimerInterrupt_HDI (void) `[inline]`

Disables the Timer2 interrupt

Disables the Timer2 interrupt and sets the interrupt flag to 0

Definition at line 132 of file system_Timer_HDI.c.

8.35.2.4 void Sys_Enable_TimerInterrupt_HDI (void) `[inline]`

Enables the Timer2 interrupt

Enables the Timer2 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 143 of file system_Timer_HDI.c.

8.35.2.5 void Sys_Force_TimerInterrupt_HDI (void) `[inline]`

forces the Timer2 interrupt

forces the Timer2 interrupt.

Definition at line 153 of file system_Timer_HDI.c.

8.35.2.6 void Sys_Init_SystemTimer_HDI (pFunction *scheduler*)

Function to initialise the system timer

This Function sets the Timer0 of the DSPIC 30F6014A for timer interfvals of 10 ms. The timer will be startet with Start_SystemTimer_HDI()

Parameters

in,out	*scheduler*	This is a pointer t an callback function, which schuld becalled whenever a timer interrupt is emmitted.

Definition at line 36 of file system_Timer_HDI.c.

8.35.2.7 void Sys_Reset_SystemTimer_HDI ( ) [inline]

Resets the Timer2 value to the initial value

This Function resets the Timer2 value

Definition at line 102 of file system_Timer_HDI.c.

8.35.2.8 void Sys_Start_SystemTimer_HDI (void)

Function to starts the initialised system timer

This Function starts the Timer2 of the DSPIC 30F6014A for timer interfvals of 10 ms. The MUST be initialised first with Init_SystemTimer_HDI()

Definition at line 62 of file system_Timer_HDI.c.

8.35.2.9 void Sys_Stop_SystemTimer_HDI ( ) [inline]

Activates the Timer2 Interrupt

This Function activated the Timer2 Interrupt

Definition at line 76 of file system_Timer_HDI.c.

8.35.3 Variable Documentation

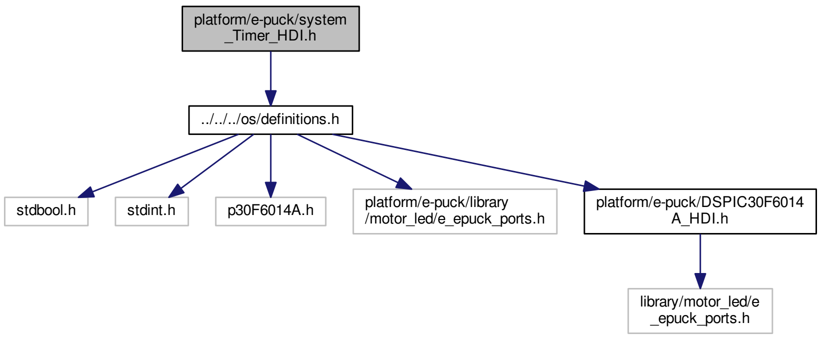
8.35.3.1 pFunction sys_process_scheduler = 0

Definition at line 27 of file system_Timer_HDI.c.

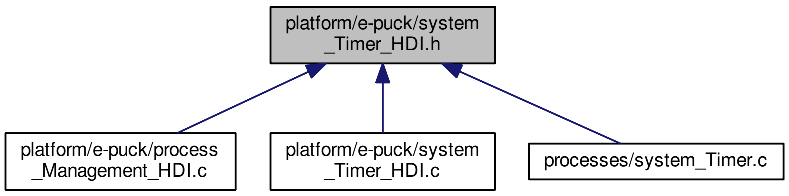
8.36 platform/e-puck/system_Timer_HDI.h File Reference

Hardware dependent implementations to initialise, configure and the operating system.

`#include "../../../os/definitions.h"`
Include dependency graph for system_Timer_HDI.h:



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



Functions

- void Sys_Init_SystemTimer_HDI (pFunction)
- void Sys_Start_SystemTimer_HDI (void)
- void Sys_Stop_SystemTimer_HDI ()
- void Sys_Continue_SystemTimer_HDI ()
- void Sys_Disable_TimerInterrupt_HDI (void)
- void Sys_Enable_TimerInterrupt_HDI (void)
- void Sys_Force_TimerInterrupt_HDI (void)
- void Sys_Reset_SystemTimer_HDI ()
- void Sys_todo_SystemTimer ()

Variables

- pFunction sys_process_scheduler

8.36.1 Detailed Description

Hardware dependent implementations to initialise, configure and the operating system.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

07 July 2014

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.36.2 Function Documentation

8.36.2.1 void Sys_Continue_SystemTimer_HDI ( ) `[inline]`

Deactivates the Timer2 Interrupt

This Function deactivated the Timer2 Interrupt

Definition at line 89 of file system_Timer_HDI.c.

8.36.2.2 void Sys_Disable_TimerInterrupt_HDI (void) `[inline]`

Disables the Timer2 interrupt

Disables the Timer2 interrupt and sets the interrupt flag to 0

Definition at line 132 of file system_Timer_HDI.c.

8.36.2.3 void Sys_Enable_TimerInterrupt_HDI (void) `[inline]`

Enables the Timer2 interrupt

Enables the Timer2 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 143 of file system_Timer_HDI.c.

8.36.2.4 void Sys_Force_TimerInterrupt_HDI (void) `[inline]`

forces the Timer2 interrupt

forces the Timer2 interrupt.

Definition at line 153 of file system_Timer_HDI.c.

8.36.2.5 void Sys_Init_SystemTimer_HDI (pFunction *scheduler*)

Function to initialise the system timer

This Function sets the Timer0 of the DSPIC 30F6014A for timer interfvals of 10 ms. The timer will be startet with Start_SystemTimer_HDI()

Parameters

in,out	*scheduler*	This is a pointer t an callback function, which schuld becalled whenever a timer interrupt is emmitted.

Definition at line 36 of file system_Timer_HDI.c.

8.36.2.6 void Sys_Reset_SystemTimer_HDI ( ) [inline]

Resets the Timer2 value to the initial value

This Function resets the Timer2 value

Definition at line 102 of file system_Timer_HDI.c.

8.36.2.7 void Sys_Start_SystemTimer_HDI (void)

Function to starts the initialised system timer

This Function starts the Timer2 of the DSPIC 30F6014A for timer interfvals of 10 ms. The MUST be initialised first with Init_SystemTimer_HDI()

Definition at line 62 of file system_Timer_HDI.c.

8.36.2.8 void Sys_Stop_SystemTimer_HDI ( ) [inline]

Activates the Timer2 Interrupt

This Function activated the Timer2 Interrupt

Definition at line 76 of file system_Timer_HDI.c.

8.36.2.9 void Sys_todo_SystemTimer ( ) [inline]

This function is executed periodically by the system timer interrupt

This function is executed periodically by the system timer interrupt. It kills all zombies, executes event handlers and executes the scheduling algorithm.

Definition at line 79 of file system_Timer.c.

8.36.3 Variable Documentation

8.36.3.1 pFunction sys_process_scheduler

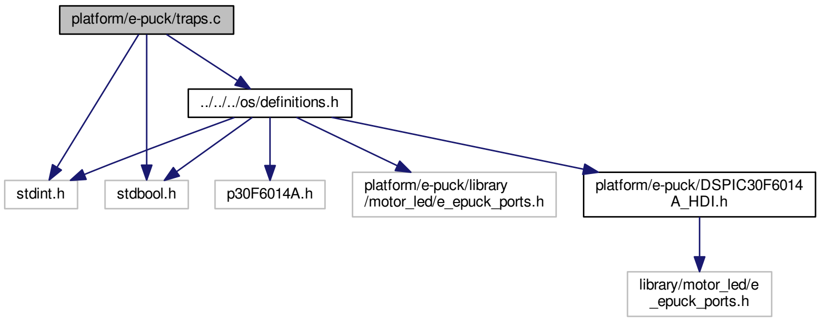
Definition at line 27 of file system_Timer_HDI.c.

8.37 platform/e-puck/traps.c File Reference

Hardware dependent implementations to catch hardware traps.

```
#include <stdint.h>
#include <stdbool.h>
#include "../../../os/definitions.h"
```

Include dependency graph for traps.c:


platform/e-puck/traps.c
../../../os/definitions.h
stdint.h
stdbool.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

Functions

- void __attribute__ ((interrupt, no_auto_psv))

8.37.1 Detailed Description

Hardware dependent implementations to catch hardware traps.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

07 July 2014

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.37.2 Function Documentation

8.37.2.1 void __attribute__ ((interrupt, no_auto_psv))

Address error trap.

This function is called when an address error occurs. That means that a call address of a function or in the stack addresses an area outside the memory. Similarly, if a pointer points to memory outside the range, this trap happens.

Stack error trap.

This function is called when an stack error occurs. That means that the stack pointer, stack pointer limit, or frame pointer are pointing outside their range.

Math error trap.

This function is called when an math error occurs. That means an illegal math operation was performed (such as division by 0 or NaN).

Alternative Oscillator fail trap.

This function is called when an oscillator fail occurs. This should never happen.

Alternative address error trap.

This function is called when an address error occurs. That means that a call address of a function or in the stack addresses an area outside the memory. Similarly, if a pointer points to memory outside the range, this trap happens.

Alternative stack error trap.

This function is called when an stack error occurs. That means that the stack pointer, stack pointer limit, or frame pointer are pointing outside their range.

Alternative math error trap.

This function is called when an math error occurs. That means an illegal math operation was performed (such as division by 0 or NaN).

Default interrupt service routine.

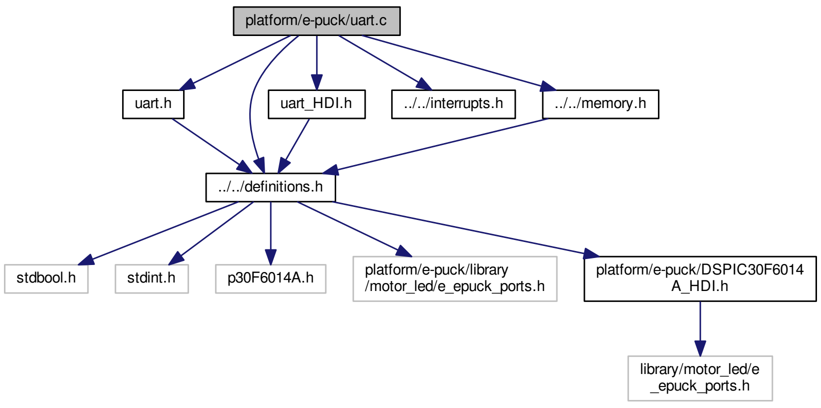
This function is called when no other interrupt routine is specified.

Definition at line 68 of file traps.c.

8.38 platform/e-puck/uart.c File Reference

This file includes functions needed to transmit data via uart(1 & 2).

```
#include "uart.h"
#include "uart_HDI.h"
#include "../../definitions.h"
#include "../../interrupts.h"
#include "../../memory.h"
```
Include dependency graph for uart.c:


platform/e-puck/uart.c
uart.h
uart_HDI.h
../../interrupts.h
../../memory.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

Macros

- #define SYS_UART1_BAUDRATE 115000
- #define SYS_UART2_BAUDRATE 115000

Functions

- void Sys_Init_UART1 (void)
- void Sys_Init_UART2 (void)
- void Sys_Start_UART1 (void)
- void Sys_Start_UART2 (void)
- void Sys_SetReadingFunction_UART1 (pUART_reader func)
- void Sys_SetReadingFunction_UART2 (pUART_reader func)
- void Sys_Writeto_UART1 (void ∗data, uint16 length)
- void Sys_Writeto_UART2 (void ∗data, uint16 length)

8.38.1 Detailed Description

This file includes functions needed to transmit data via uart(1 & 2).

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.38.2 Macro Definition Documentation

8.38.2.1 #define SYS_UART1_BAUDRATE 115000

Baudrate for UART 1 (bits/s)

Definition at line 24 of file uart.c.

8.38.2.2 #define SYS_UART2_BAUDRATE 115000

Baudrate for UART 2 (bits/s)

Definition at line 25 of file uart.c.

8.38.3 Function Documentation

8.38.3.1 void Sys_Init_UART1 (void) [inline]

Initialises UART1

This function initialises UART1.

Definition at line 34 of file uart.c.

8.38.3.2 void Sys_Init_UART2 (void) `[inline]`

Initialises UART2

This function initialises UART2.

Definition at line 44 of file uart.c.

8.38.3.3 void Sys_SetReadingFunction_UART1 (pUART_reader *func*)

defines a function that processes received bytes (UART1)

defines a function that processes received bytes (UART1). This defined callback function is only executed once by arrival of one byte.

Parameters

`in`	*func*	pointer to the function that should process the received byte(s).

Definition at line 79 of file uart.c.

8.38.3.4 void Sys_SetReadingFunction_UART2 (pUART_reader *func*)

defines a function that processes received bytes (UART2)

defines a function that processes received bytes (UART2). This defined callback function is only executed once by arrival of one byte.

Parameters

`in`	*func*	pointer to the function that should process the received byte(s).

Definition at line 90 of file uart.c.

8.38.3.5 void Sys_Start_UART1 (void) `[inline]`

starts UART1

This function starts UART1.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 56 of file uart.c.

8.38.3.6 void Sys_Start_UART2 (void) `[inline]`

starts UART2

This function starts UART2.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 68 of file uart.c.

8.38.3.7 void Sys_Writeto_UART1 (void ∗ *data*, uint16 *length*)

writes a set of bytes to UART1

This function writes sequentially the bytes on the UART1.

Note

The data will be put into a queue, where it will be sent as soon as the UART is idle

Parameters

`in`	*data*	pointer to the bytes that should be transmitted.
`in`	*length*	number of bytes to send.

Definition at line 104 of file uart.c.

8.38.3.8 void Sys_Writeto_UART2 (void ∗ *data,* uint16 *length*)

writes a set of bytes to UART2

This function writes sequentially the bytes on the UART2.

Note

The data will be put into a queue, where it will be sent as soon as the UART is idle

Parameters

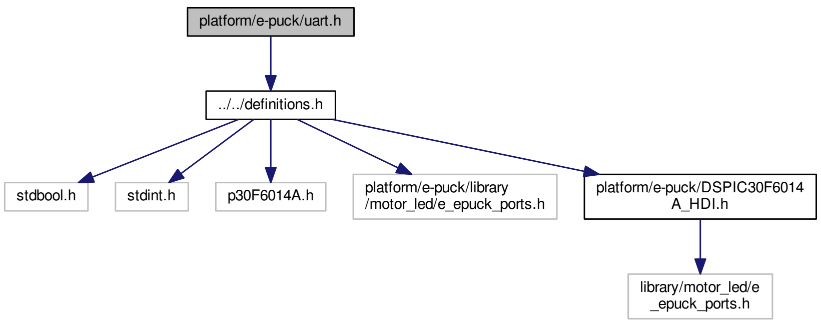
`in`	*data*	pointer to the bytes that should be transmitted.
`in`	*length*	number of bytes to send.

Definition at line 144 of file uart.c.

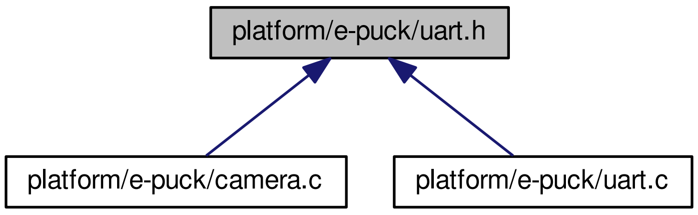
8.39 platform/e-puck/uart.h File Reference

This file includes functions needed to transmit data via uart(1 & 2).

`#include "../../definitions.h"`
Include dependency graph for uart.h:


platform/e-puck/uart.h
../../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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



Functions

- void Sys_Init_UART1 (void)
- void Sys_Init_UART2 (void)
- void Sys_Start_UART1 (void)
- void Sys_Start_UART2 (void)
- void Sys_SetReadingFunction_UART1 (pUART_reader func)
- void Sys_SetReadingFunction_UART2 (pUART_reader func)
- void Sys_Writeto_UART1 (void ∗data, uint16 length)
- void Sys_Writeto_UART2 (void ∗data, uint16 length)

8.39.1 Detailed Description

This file includes functions needed to transmit data via uart(1 & 2).

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.39.2 Function Documentation

8.39.2.1 void Sys_Init_UART1 (void) [inline]

Initialises UART1

This function initialises UART1.

Definition at line 34 of file uart.c.

8.39.2.2 void Sys_Init_UART2 (void) [inline]

Initialises UART2

This function initialises UART2.

Definition at line 44 of file uart.c.

8.39.2.3 void Sys_SetReadingFunction_UART1 (pUART_reader *func*)

defines a function that processes received bytes (UART1)

defines a function that processes received bytes (UART1). This defined callback function is only executed once by arrival of one byte.

Parameters

in	*func*	pointer to the function that should process the received byte(s).

Definition at line 79 of file uart.c.

8.39.2.4 void Sys_SetReadingFunction_UART2 (pUART_reader *func*)

defines a function that processes received bytes (UART2)

defines a function that processes received bytes (UART2). This defined callback function is only executed once by arrival of one byte.

Parameters

in	*func*	pointer to the function that should process the received byte(s).

Definition at line 90 of file uart.c.

8.39.2.5 void Sys_Start_UART1 (void) [inline]

starts UART1

This function starts UART1.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 56 of file uart.c.

8.39.2.6 void Sys_Start_UART2 (void) [inline]

starts UART2

This function starts UART2.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 68 of file uart.c.

8.39.2.7 void Sys_Writeto_UART1 (void $*$ *data*, uint16 *length*)

writes a set of bytes to UART1

This function writes sequentially the bytes on the UART1.

Note

The data will be put into a queue, where it will be sent as soon as the UART is idle

Parameters

in	*data*	pointer to the bytes that should be transmitted.
in	*length*	number of bytes to send.

Definition at line 104 of file uart.c.

8.39.2.8 void Sys_Writeto_UART2 (void ∗ *data,* uint16 *length*)

writes a set of bytes to UART2

This function writes sequentially the bytes on the UART2.

Note

The data will be put into a queue, where it will be sent as soon as the UART is idle

Parameters

in	*data*	pointer to the bytes that should be transmitted.
in	*length*	number of bytes to send.

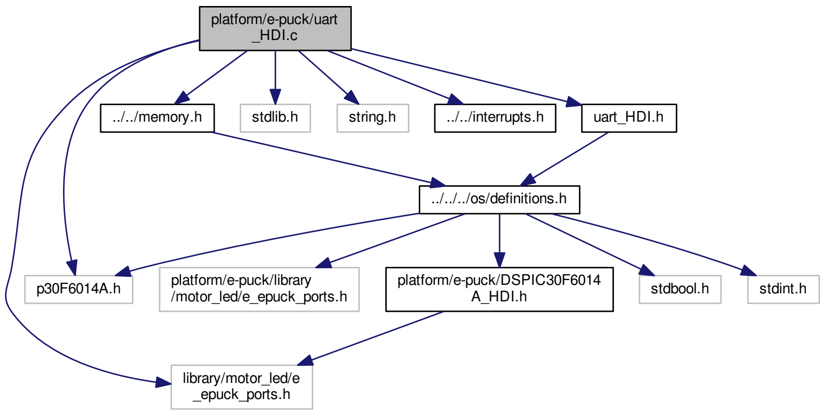
Definition at line 144 of file uart.c.

8.40 platform/e-puck/uart_HDI.c File Reference

Hardware dependent implementations to control the message flow of the UART interface.

```
#include "uart_HDI.h"
#include <p30F6014A.h>
#include <stdlib.h>
#include <string.h>
#include "library/motor_led/e_epuck_ports.h"
#include "../../interrupts.h"
#include "../../memory.h"
```

Include dependency graph for uart_HDI.c:


platform/e-puck/uart_HDI.c
../../memory.h
stdlib.h
string.h
../../interrupts.h
uart_HDI.h
../../../os/definitions.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
stdbool.h
stdint.h
library/motor_led/e_epuck_ports.h

Functions

- void Sys_Init_UART1_HDI (void)
- void Sys_Init_UART2_HDI (void)
- void Sys_Start_UART1_HDI (void)
- void Sys_Start_UART2_HDI (void)
- void __attribute__ ((interrupt, auto_psv))
- void Sys_Read_UART1_ISR ()
- void Sys_Write_UART1_ISR ()
- void Sys_Read_UART2_ISR ()
- void Sys_Write_UART2_ISR ()

Variables

- pUART_reader read_uart_1 = 0
- pUART_reader read_uart_2 = 0
- sys_uart_txdata ∗ sys_UART1_TX_data = 0
- sys_uart_txdata ∗ sys_UART2_TX_data = 0
- uint16 byte_counter_uart1 = 0
- uint16 byte_counter_uart2 = 0

8.40.1 Detailed Description

Hardware dependent implementations to control the message flow of the UART interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.40.2 Function Documentation

8.40.2.1 void __attribute__ ((interrupt, auto_psv))

UART1 reading interrupt

UART1 reading interrupt.

Alternative UART1 reading interrupt

Alternative UART1 reading interrupt.

UART1 writing interrupt

UART1 writing interrupt.

Alternative UART1 writing interrupt

Alternative UART1 writing interrupt.

UART2 reading interrupt

UART2 reading interrupt.

Alternative UART2 reading interrupt

Alternative UART2 reading interrupt.

UART2 writing interrupt

UART2 writing interrupt.

Alternative UART2 writing interrupt

Alternative UART2 writing interrupt.

Definition at line 143 of file uart_HDI.c.

8.40.2.2 void Sys_Init_UART1_HDI (void)

Initialises UART1

This function initialises UART1.

Definition at line 45 of file uart_HDI.c.

8.40.2.3 void Sys_Init_UART2_HDI (void)

Initialises UART2

This function initialises UART2.

Definition at line 83 of file uart_HDI.c.

8.40.2.4 void Sys_Read_UART1_ISR ( ) `[inline]`

UART1 reading function

This function is executed at occurrence of the UART1 reading interrupt.

Definition at line 228 of file uart_HDI.c.

8.40.2.5 void Sys_Read_UART2_ISR ( ) `[inline]`

UART2 reading function

This function is executed at occurrence of the UART2 reading interrupt.

Definition at line 288 of file uart_HDI.c.

8.40.2.6 void Sys_Start_UART1_HDI (void)

starts UART2

This function starts UART2.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 111 of file uart_HDI.c.

8.40.2.7 void Sys_Start_UART2_HDI (void)

starts UART2

This function starts UART2.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 128 of file uart_HDI.c.

8.40.2.8 void Sys_Write_UART1_ISR ( ) `[inline]`

UART1 writing function

This function is executed at occurrence of the UART1 writing interrupt.

Definition at line 249 of file uart_HDI.c.

8.40.2.9 void Sys_Write_UART2_ISR ( ) `[inline]`

UART1 writing function

This function is executed at occurrence of the UART1 writing interrupt.

Definition at line 309 of file uart_HDI.c.

8.40.3 Variable Documentation

8.40.3.1 uint16 byte_counter_uart1 = 0

Bytes that were written

Definition at line 36 of file uart_HDI.c.

8.40.3.2 uint16 byte_counter_uart2 = 0

Bytes that were written

Definition at line 37 of file uart_HDI.c.

8.40.3.3 pUART_reader read_uart_1 = 0

pointer to the functions that processes incoming bytes from UART1

Definition at line 30 of file uart_HDI.c.

8.40.3.4 pUART_reader read_uart_2 = 0

pointer to the functions that processes incoming bytes from UART2

Definition at line 31 of file uart_HDI.c.

8.40.3.5 sys_uart_txdata∗ sys_UART1_TX_data = 0

Linked list of messages that need to be sent via UART1

Definition at line 33 of file uart_HDI.c.

8.40.3.6 sys_uart_txdata∗ sys_UART2_TX_data = 0

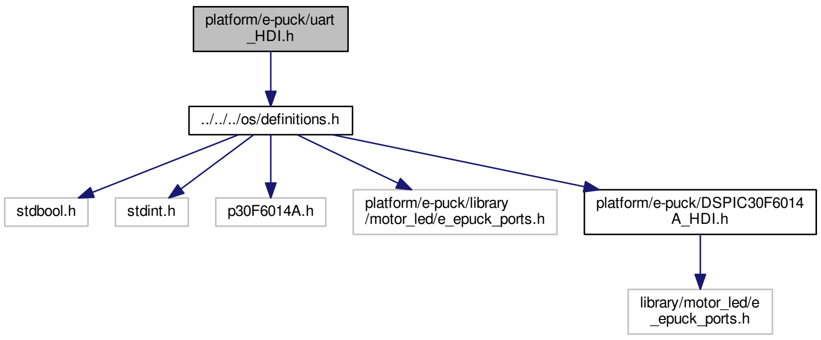
Linked list of messages that need to be sent via UART2

Definition at line 34 of file uart_HDI.c.

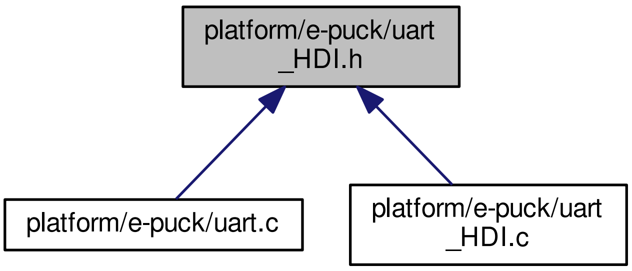
8.41 platform/e-puck/uart_HDI.h File Reference

Hardware dependent implementations to control the message flow of the UART interface.

`#include "../../../os/definitions.h"`
Include dependency graph for uart_HDI.h:


platform/e-puck/uart _HDI.h
../../../os/definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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



Data Structures

- struct uart_tx_data_s

Macros

- #define UART1_RX _RF2
- #define UART1_TX _RF3

- #define UART2_RX _RF4
- #define UART2_TX _RF5
- #define UART1_RX_DIR _TRISF2
- #define UART1_TX_DIR _TRISF3
- #define UART2_RX_DIR _TRISF4
- #define UART2_TX_DIR _TRISF5
- #define SYS_UART1_BAUDRATE 115000
- #define SYS_UART2_BAUDRATE 115000

Typedefs

- typedef struct uart_tx_data_s sys_uart_txdata

Functions

- void Sys_Init_UART1_HDI (void)
- void Sys_Init_UART2_HDI (void)
- void Sys_Start_UART1_HDI (void)
- void Sys_Start_UART2_HDI (void)
- void Sys_Read_UART1_ISR ()
- void Sys_Write_UART1_ISR ()
- void Sys_Read_UART2_ISR ()
- void Sys_Write_UART2_ISR ()

Variables

- sys_uart_txdata ∗ sys_UART1_TX_data
- sys_uart_txdata ∗ sys_UART2_TX_data
- uint16 byte_counter_uart1
- uint16 byte_counter_uart2
- pUART_reader read_uart_1
- pUART_reader read_uart_2

8.41.1 Detailed Description

Hardware dependent implementations to control the message flow of the UART interface.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

27 August 2015

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.41.2 Macro Definition Documentation

8.41.2.1 #define SYS_UART1_BAUDRATE 115000

Baud rate for UART1

Definition at line 38 of file uart_HDI.h.

8.41.2.2 #define SYS_UART2_BAUDRATE 115000

Baud rate for UART2

Definition at line 39 of file uart_HDI.h.

8.41.2.3 #define UART1_RX _RF2

Used port on the microcontroller to read from the UART1

Definition at line 28 of file uart_HDI.h.

8.41.2.4 #define UART1_RX_DIR _TRISF2

direction of the used port on the microcontroller (reading from the UART1)

Definition at line 33 of file uart_HDI.h.

8.41.2.5 #define UART1_TX _RF3

Used port on the microcontroller to write on the UART1

Definition at line 29 of file uart_HDI.h.

8.41.2.6 #define UART1_TX_DIR _TRISF3

direction of the used port on the microcontroller (writing from the UART1)

Definition at line 34 of file uart_HDI.h.

8.41.2.7 #define UART2_RX _RF4

Used port on the microcontroller to read from the UART2

Definition at line 30 of file uart_HDI.h.

8.41.2.8 #define UART2_RX_DIR _TRISF4

direction of the used port on the microcontroller (reading from the UART2)

Definition at line 35 of file uart_HDI.h.

8.41.2.9 #define UART2_TX _RF5

Used port on the microcontroller to write on the UART2

Definition at line 31 of file uart_HDI.h.

8.41.2.10 #define UART2_TX_DIR _TRISF5

direction of the used port on the microcontroller (writing from the UART2)

Definition at line 36 of file uart_HDI.h.

8.41.3 Typedef Documentation

8.41.3.1 typedef struct uart_tx_data_s sys_uart_txdata

Linked list element to store the transmission data

This struct contains data and the amount of bytes that should be sent via UART1.

8.41.4 Function Documentation

8.41.4.1 void Sys_Init_UART1_HDI (void)

Initialises UART1

This function initialises UART1.

Definition at line 45 of file uart_HDI.c.

8.41.4.2 void Sys_Init_UART2_HDI (void)

Initialises UART2

This function initialises UART2.

Definition at line 83 of file uart_HDI.c.

8.41.4.3 void Sys_Read_UART1_ISR ( ) [inline]

UART1 reading function

This function is executed at occurrence of the UART1 reading interrupt.

Definition at line 228 of file uart_HDI.c.

8.41.4.4 void Sys_Read_UART2_ISR ( ) [inline]

UART2 reading function

This function is executed at occurrence of the UART2 reading interrupt.

Definition at line 288 of file uart_HDI.c.

8.41.4.5 void Sys_Start_UART1_HDI (void)

starts UART2

This function starts UART2.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 111 of file uart_HDI.c.

8.41.4.6 void Sys_Start_UART2_HDI (void)

starts UART2

This function starts UART2.

Note

When executed this function, bytes can be received or transmitted at any time.

Definition at line 128 of file uart_HDI.c.

8.41.4.7 void Sys_Write_UART1_ISR ( ) [inline]

UART1 writing function

This function is executed at occurrence of the UART1 writing interrupt.

Definition at line 249 of file uart_HDI.c.

8.41.4.8 void Sys_Write_UART2_ISR ( ) [inline]

UART1 writing function

This function is executed at occurrence of the UART1 writing interrupt.

Definition at line 309 of file uart_HDI.c.

8.41.5 Variable Documentation

8.41.5.1 uint16 byte_counter_uart1

Bytes that were written

Definition at line 36 of file uart_HDI.c.

8.41.5.2 uint16 byte_counter_uart2

Bytes that were written

Definition at line 37 of file uart_HDI.c.

8.41.5.3 pUART_reader read_uart_1

pointer to the functions that processes incoming bytes from UART1

Definition at line 30 of file uart_HDI.c.

8.41.5.4 pUART_reader read_uart_2

pointer to the functions that processes incoming bytes from UART2

Definition at line 31 of file uart_HDI.c.

8.41.5.5 sys_uart_txdata$*$ sys_UART1_TX_data

Linked list of messages that need to be sent via UART1

Definition at line 33 of file uart_HDI.c.

8.41.5.6 sys_uart_txdata∗ sys_UART2_TX_data

Linked list of messages that need to be sent via UART2

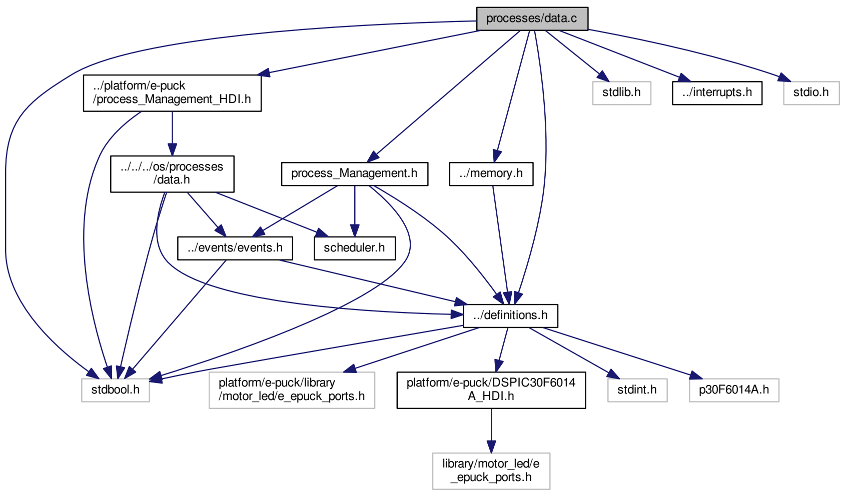
Definition at line 34 of file uart_HDI.c.

8.42 processes/data.c File Reference

This file includes all functions which are needed to manage data structures needed by the processes management.

```
#include "process_Management.h"
#include "../platform/e-puck/process_Management_HDI.h"
#include <stdlib.h>
#include "../interrupts.h"
#include "../memory.h"
#include "../definitions.h"
#include <stdbool.h>
#include <stdio.h>
```

Include dependency graph for data.c:



Functions

- sys_pcb_list_element ∗ Sys_Remove_Process_from_List (uint16 pID, sys_pcb_list_element ∗∗list)
- sys_pcb_list_element ∗ Sys_Find_Process (uint16 pid)
- sys_process_event_handler ∗ Sys_Next_EventHandler (sys_process_event_handler ∗list, uint16 eventID)
- sys_process_event_handler ∗ Sys_Remove_Event_from_EventRegister (uint16 eventID, pEventHandler←Function func, sys_process_event_handler ∗∗list)
- void Sys_Clear_EventData (sys_event_data ∗∗data)
- void Sys_Clear_EventRegister (sys_pcb_list_element ∗element)
- void Sys_Delete_Process (sys_pcb_list_element ∗element)
- bool Sys_Set_Defaults_PCB (sys_pcb ∗element, uint16 stacksize)
- void Sys_Insert_Process_to_List (sys_pcb_list_element ∗process, sys_pcb_list_element ∗∗list)

Variables

- sys_pcb_list_element ∗ sys_ready_processes = 0

- sys_pcb_list_element ∗ sys_running_process = 0
- sys_pcb_list_element ∗ sys_blocked_processes = 0
- sys_pcb_list_element ∗ sys_zombies = 0
- sys_occured_event ∗ sys_occurred_events = 0

8.42.1 Detailed Description

This file includes all functions which are needed to manage data structures needed by the processes management.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

08 July 2014

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.42.2 Function Documentation

8.42.2.1 void Sys_Clear_EventData (**sys_event_data** ∗∗ *data*) `[inline]`

This function removes and frees a list of sys_event_data

This function removes and frees a list of sys_event_data

Parameters

`in,out`	*data*	pointer to the event_data (list)

Returns

void

Definition at line 219 of file data.c.

8.42.2.2 void Sys_Clear_EventRegister (**sys_pcb_list_element** ∗ *element*) `[inline]`

This function clears and frees all elements of a process

This function clears and frees all elements of a process. The process is also unsubscribed from any event, because and empty event register cannot handle any events.

Parameters

`in,out`	*element*	pointer to the pcb of the process

Returns

void

Definition at line 241 of file data.c.

8.42.2.3 void Sys_Delete_Process (**sys_pcb_list_element** ∗ *element*)

This function deletes container elements

This function deletes container elements. Warning: this function only deletes the process. All the elements which are linked with next are lost in memory, if you haven't take care of that on advance.

Parameters

`in`	*element*	pointer to the element which should be deleted

Returns

void

Definition at line 264 of file data.c.

8.42.2.4 **sys_pcb_list_element**∗ Sys_Find_Process (**uint16** *pid*) `[inline]`

This function return the pointer to the PCB of process with pid

This function return the pointer to the PCB of process with pid

Parameters

`in`	*pid*	process ID

Returns

void

Definition at line 108 of file data.c.

8.42.2.5 void Sys_Insert_Process_to_List (**sys_pcb_list_element** ∗ *process,* **sys_pcb_list_element** ∗∗ *list*)

This function inserts a process into a process list.

This function inserts a process into a process list. Note: The elements are sorted (process ID)

Parameters

`in`	*process*	the process struct
`in,out`	∗∗*list*	the process list which has to be seached

Returns

void

Definition at line 318 of file data.c.

8.42.2.6 **sys_process_event_handler**∗ Sys_Next_EventHandler (**sys_process_event_handler** ∗ *list,* **uint16** *eventID*) `[inline]`

This function searches (sequentially) all event handler for an event (eventID)

This function searches (sequentially) all event handler for an event (eventID). The list contains a list of eventhandler and this function return the first occurrence of eventID. To search the list entirely, use the function on a list and after resulting an element use the same function on the next element (sublist).

Parameters

`in`	*list*	list of event handler
`in`	*eventID*	The Id of the event which can put the process (PID) back on the ready list

Returns

sys_process_event_handler ∗ pointer to the next event handler for the event (eventID) in list (0 if not found)

Definition at line 145 of file data.c.

8.42.2.7 sys_process_event_handler∗ Sys_Remove_Event_from_EventRegister (uint16 *eventID,* pEventHandlerFunction *func,* sys_process_event_handler ∗∗ *list*) `[inline]`

This function removes subscribed handler function from event-handler list

This function removes subscribed handler function from event-handler list

Parameters

`in`	*eventID*	Identifier of the event that has to be removed
`in`	*func*	pointer to the subscribed handler function
`in`	*list*	list of event handlers

Returns

sys_process_event_handler ∗ (New) top of the list (if changed)

Definition at line 174 of file data.c.

8.42.2.8 sys_pcb_list_element∗ Sys_Remove_Process_from_List (uint16 *pID,* sys_pcb_list_element ∗∗ *list*)

This function removed a pcb element from the list

This function seaches all elements of a process list and removes the processs pID from it. Note: The element is not deleted. The pointer to it is returned.

Parameters

`in`	*pID*	the process identifier
`in,out`	∗∗*list*	the process list which has to be seached

Returns

sys_pcb_list_element∗ the pointer to the removed element

Definition at line 53 of file data.c.

8.42.2.9 bool Sys_Set_Defaults_PCB (sys_pcb ∗ *element,* uint16 *stacksize*) `[inline]`

This function sets default values to the sys_process_control_block struct

This function sets the default values in a sys_process_control_block struct

Parameters

`in,out`	*element*	This is a pointer to a sys_process_control_block struct
`in`	*stacksize*	This is a uint16 whch represents the size of the stack which should be allocated for this process. The default value (=0) is in DEFAULT_PROCESS_STACK←_SIZE.

Returns

void

Definition at line 285 of file data.c.

8.42.3 Variable Documentation

8.42.3.1 sys_pcb_list_element∗ sys_blocked_processes = 0

pointer to the blocked process

Definition at line 34 of file data.c.

8.42.3.2 sys_occured_event∗ sys_occurred_events = 0

pointer to the occurred events

Definition at line 36 of file data.c.

8.42.3.3 sys_pcb_list_element∗ sys_ready_processes = 0

pointer to the ready processes (linked list)

Definition at line 32 of file data.c.

8.42.3.4 sys_pcb_list_element∗ sys_running_process = 0

pointer to the running process

Definition at line 33 of file data.c.

8.42.3.5 sys_pcb_list_element∗ sys_zombies = 0

pointer to the zombie process

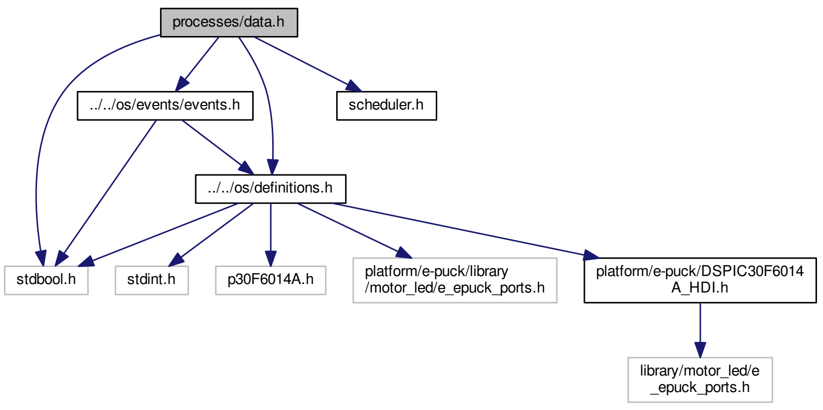
Definition at line 35 of file data.c.

8.43 processes/data.h File Reference

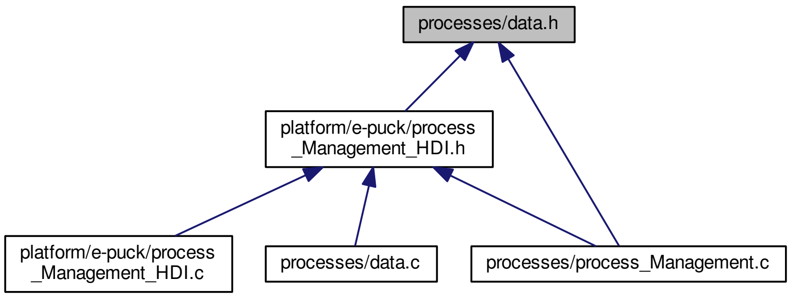
This file includes all functions which are needed to manage data structures needed by the processes management.

```
#include <stdbool.h>
#include "../../os/definitions.h"
#include "../../os/events/events.h"
#include "scheduler.h"
```

Include dependency graph for data.h:


processes/data.h
../../os/events/events.h
scheduler.h
../../os/definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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


processes/data.h
platform/e-puck/process _Management_HDI.h
platform/e-puck/process _Management_HDI.c
processes/data.c
processes/process_Management.c

Data Structures

- struct sys_occured_event_s

  List of occured events.
- struct sys_process_event_handler_s

  List of process event-handlers.
- struct sys_process_control_block_s

  Process Control Block for the processes.
- struct sys_process_control_block_list_element_s

  Container struct for Process Control Block.

Typedefs

- typedef struct sys_occured_event_s sys_occured_event

  List of occured events.
- typedef struct sys_process_event_handler_s sys_process_event_handler

  List of process event-handlers.

- typedef struct sys_process_event_handler_s sys_peh
- typedef struct sys_process_control_block_s sys_process_control_block

  Process Control Block for the processes.
- typedef struct sys_process_control_block_s sys_pcb
- typedef struct sys_process_control_block_list_element_s sys_process_control_block_list_element

  Container struct for Process Control Block.
- typedef struct sys_process_control_block_list_element_s sys_pcb_list_element

Functions

- sys_pcb_list_element ∗ Sys_Find_Process (uint16 pid)
- sys_pcb_list_element ∗ Sys_Remove_Process_from_List (uint16 pID, sys_pcb_list_element ∗∗list)
- void Sys_Delete_Process (sys_pcb_list_element ∗element)
- bool Sys_Set_Defaults_PCB (sys_process_control_block ∗element, uint16 stacksize)
- void Sys_Insert_Process_to_List (sys_pcb_list_element ∗process, sys_pcb_list_element ∗∗list)
- sys_process_event_handler ∗ Sys_Next_EventHandler (sys_process_event_handler ∗list, uint16 eventID)
- void Sys_Clear_EventRegister (sys_pcb_list_element ∗element)
- void Sys_Clear_EventData (sys_event_data ∗∗data)
- sys_process_event_handler ∗ Sys_Find_EventHandler (sys_process_event_handler ∗process, uint16 eventID)
- sys_process_event_handler ∗ Sys_Remove_Event_from_EventRegister (uint16 eventID, pEventHandler←Function func, sys_process_event_handler ∗∗list)

Variables

- sys_pcb_list_element ∗ sys_ready_processes
- sys_pcb_list_element ∗ sys_running_process
- sys_pcb_list_element ∗ sys_blocked_processes
- sys_pcb_list_element ∗ sys_zombies
- sys_occured_event ∗ sys_occurred_events

8.43.1 Detailed Description

This file includes all functions which are needed to manage data structures needed by the processes management.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

08 July 2014

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.43.2 Typedef Documentation

8.43.2.1 typedef struct sys_occured_event_s sys_occured_event

List of occured events.

This struct sores the event ID of an occurred event

8.43.2.2 typedef struct sys_process_control_block_s sys_pcb

8.43.2.3 typedef struct sys_process_control_block_list_element_s sys_pcb_list_element

8.43.2.4 typedef struct sys_process_event_handler_s sys_peh

8.43.2.5 typedef struct sys_process_control_block_s sys_process_control_block

Process Control Block for the processes.

This struct sores all information of the current state of a process

8.43.2.6 typedef struct sys_process_control_block_list_element_s sys_process_control_block_list_element

Container struct for Process Control Block.

This struct is a container (linked list) for PCB

8.43.2.7 typedef struct sys_process_event_handler_s sys_process_event_handler

List of process event-handlers.

This struct sores all information needed to decide if the event-handler is executed for the event (eventID). To store the event data and be executed, a condition has to be met.

8.43.3 Function Documentation

8.43.3.1 void Sys_Clear_EventData (sys_event_data ** *data*) `[inline]`

This function removes and frees a list of sys_event_data

This function removes and frees a list of sys_event_data

Parameters

`in,out`	*data*	pointer to the event_data (list)

Returns

void

Definition at line 219 of file data.c.

8.43.3.2 void Sys_Clear_EventRegister (sys_pcb_list_element * *element*) `[inline]`

This function clears and frees all elements of a process

This function clears and frees all elements of a process. The process is also unsubscribed from any event, because and empty event register cannot handle any events.

Parameters

`in,out`	*element*	pointer to the pcb of the process

Returns

void

Definition at line 241 of file data.c.

8.43.3.3 void Sys_Delete_Process (sys_pcb_list_element ∗ *element*)

This function deletes container elements

This function deletes container elements. Warning: this function only deletes the process. All the elements which are linked with next are lost in memory, if you haven't take care of that on advance.

Parameters

`in`	*element*	pointer to the element which should be deleted

Returns

void

Definition at line 264 of file data.c.

8.43.3.4 sys_process_event_handler∗ Sys_Find_EventHandler (sys_process_event_handler ∗ *process,* uint16 *eventID*) `[inline]`

8.43.3.5 sys_pcb_list_element∗ Sys_Find_Process (uint16 *pid*) `[inline]`

This function return the pointer to the PCB of process with pid

This function return the pointer to the PCB of process with pid

Parameters

`in`	*pid*	process ID

Returns

void

Definition at line 108 of file data.c.

8.43.3.6 void Sys_Insert_Process_to_List (sys_pcb_list_element ∗ *process,* sys_pcb_list_element ∗∗ *list*)

This function inserts a process into a process list.

This function inserts a process into a process list. Note: The elements are sorted (process ID)

Parameters

`in`	*process*	the process struct

`in,out`	*∗∗list*	the process list which has to be seached

Returns

void

Definition at line 318 of file data.c.

8.43.3.7 sys_process_event_handler∗ Sys_Next_EventHandler (sys_process_event_handler ∗ *list,* uint16 *eventID*) `[inline]`

This function searches (sequentially) all event handler for an event (eventID)

This function searches (sequentially) all event handler for an event (eventID). The list contains a list of eventhandler and this function return the first occurrence of eventID. To search the list entirely, use the function on a list and after resulting an element use the same function on the next element (sublist).

Parameters

`in`	*list*	list of event handler
`in`	*eventID*	The Id of the event which can put the process (PID) back on the ready list

Returns

sys_process_event_handler ∗ pointer to the next event handler for the event (eventID) in list (0 if not found)

Definition at line 145 of file data.c.

8.43.3.8 sys_process_event_handler∗ Sys_Remove_Event_from_EventRegister (uint16 *eventID,* pEventHandlerFunction *func,* sys_process_event_handler ∗∗ *list*) `[inline]`

This function removes subscribed handler function from event-handler list

This function removes subscribed handler function from event-handler list

Parameters

`in`	*eventID*	Identifier of the event that has to be removed
`in`	*func*	pointer to the subscribed handler function
`in`	*list*	list of event handlers

Returns

sys_process_event_handler ∗ (New) top of the list (if changed)

Definition at line 174 of file data.c.

8.43.3.9 sys_pcb_list_element∗ Sys_Remove_Process_from_List (uint16 *pID,* sys_pcb_list_element ∗∗ *list*)

This function removed a pcb element from the list

This function seaches all elements of a process list and removes the processs pID from it. Note: The element is not deleted. The pointer to it is returned.

Parameters

in	*pID*	the process identifier
in,out	∗∗*list*	the process list which has to be seached

Returns

sys_pcb_list_element∗ the pointer to the removed element

Definition at line 53 of file data.c.

8.43.3.10 bool Sys_Set_Defaults_PCB (sys_pcb ∗ *element*, uint16 *stacksize*) [inline]

This function sets default values to the sys_process_control_block struct

This function sets the default values in a sys_process_control_block struct

Parameters

in,out	*element*	This is a pointer to a sys_process_control_block struct
in	*stacksize*	This is a uint16 whch represents the size of the stack which should be allocated for this process. The default value (=0) is in DEFAULT_PROCESS_STACK←_SIZE.

Returns

void

Definition at line 285 of file data.c.

8.43.4 Variable Documentation

8.43.4.1 sys_pcb_list_element∗ sys_blocked_processes

pointer to the blocked process

Definition at line 34 of file data.c.

8.43.4.2 sys_occured_event∗ sys_occurred_events

pointer to the occurred events

Definition at line 36 of file data.c.

8.43.4.3 sys_pcb_list_element∗ sys_ready_processes

pointer to the ready processes (linked list)

Definition at line 32 of file data.c.

8.43.4.4 sys_pcb_list_element∗ sys_running_process

pointer to the running process

Definition at line 33 of file data.c.

8.43.4.5 sys_pcb_list_element∗ sys_zombies

pointer to the zombie process

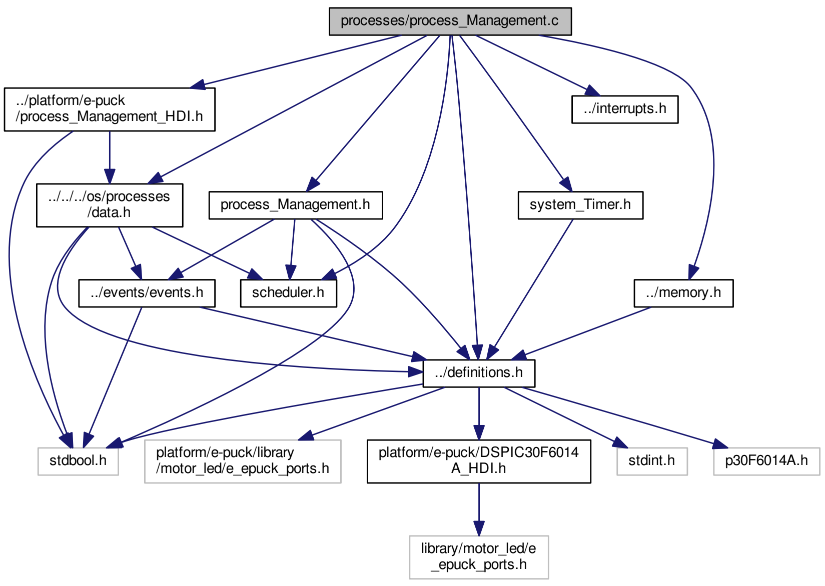
Definition at line 35 of file data.c.

8.44 processes/process_Management.c File Reference

This file includes all functions wich are needed to manage processes (e.g. task swichting)

```
#include "process_Management.h"
#include "../platform/e-puck/process_Management_HDI.h"
#include "data.h"
#include "scheduler.h"
#include "system_Timer.h"
#include "../interrupts.h"
#include "../memory.h"
#include "../definitions.h"
```

Include dependency graph for process_Management.c:



Functions

- void Sys_Block_Process (uint16 pid, uint16 eventID, pConditionFunction condition)
- bool Sys_Continue_Pocess (uint16 pid, uint16 eventID, sys_event_data ∗data)
- void Sys_Set_Running_Process_to_Zombie ()
- void Sys_Init_Process_Management ()
- unsigned short Sys_Get_Number_Processes ()
- bool Sys_Start_Process (pFunction function)
- void Sys_Kill_Process (uint16 pid)
- void Sys_Kill_Zombies ()
- void Sys_Switch_Process (uint16 pid)
- void Sys_Switch_to_next_Process ()
- void Sys_Start_CriticalSection (void)

- void Sys_End_CriticalSection (void)
- bool Sys_Add_Event_Subscription (uint16 pid, uint16 eventID, pEventHandlerFunction func, pCondition↩ Function cond)
- void Sys_Add_Event_to_Process (uint16 pid, uint16 eventID, void ∗data, uint16 length)
- void Sys_Execute_Events_in_ProcessList (uint16 eventID, sys_pcb_list_element ∗elements)
- void Sys_Execute_All_EventHandler ()
- void Sys_Interprocess_EventHandling ()
- void Sys_Remove_All_Event_Subscriptions (uint16 eventID)
- void Sys_Remove_Event_Subscription (uint16 pid, uint16 eventID, pEventHandlerFunction func)
- sys_event_data ∗ Sys_Wait_For_Condition (uint16 eventID, pConditionFunction function)
- sys_event_data ∗ Sys_Wait_For_Event (uint16 eventID)
- void Sys_Yield ()

8.44.1 Detailed Description

This file includes all functions wich are needed to manage processes (e.g. task swichting)

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

{08 July 2014}

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.44.2 Function Documentation

8.44.2.1 bool Sys_Add_Event_Subscription (uint16 *pid*, uint16 *eventID*, pEventHandlerFunction *func*, pConditionFunction *cond*)

Puts a process on the blocked list and stops its execution (if it's executed)

Puts a process on the blocked list and stops its execution (if it's executed)

Parameters

in	*pid*	Process ID
in	*eventID*	The Id of the event which can put the process (PID) back on the ready list
in	*func*	The function that handles the event
in	*cond*	The condition under which the handler is executed

Returns

bool Was the event-handler successfully added?

Definition at line 336 of file process_Management.c.

8.44.2.2 void Sys_Add_Event_to_Process (uint16 *pid*, uint16 *eventID*, void ∗ *data*, uint16 *length*)

This function adds the event-data to the local list of the process (pid).

This function adds the event-data to the local list of the process (pid).

Parameters

in	*pid*	process identifier
in	*eventID*	event identifier
in	*data*	memory that contains the value of the occurred event
in	*length*	length of the data (bytes)

Definition at line 390 of file process_Management.c.

8.44.2.3 void Sys_Block_Process (uint16 *pid,* uint16 *eventID,* pConditionFunction *condition*)

Puts a process on the blocked list and stops its execution (if it's executed)

Puts a process on the blocked list and stops its execution (if it's executed)

Parameters

in	*pid*	Process ID
in	*eventID*	The Id of the event which can put the process (PID) back on the ready list
in	*condition*	the condition under which the process is released

Definition at line 260 of file process_Management.c.

8.44.2.4 bool Sys_Continue_Pocess (uint16 *pid,* uint16 *eventID,* sys_event_data ∗ *data*)

Puts a process on the ready list

Puts a process with the process ID (PID) back on the ready list. Consequently the process can be executed again.

Parameters

in	*pid*	Process ID
in	*eventID*	Event ID
in	*data*	pointer to the data of the event

Definition at line 290 of file process_Management.c.

8.44.2.5 void Sys_End_CriticalSection (void) [inline]

Ends a critical section

This Function ends a critical section. The task-scheduling can now occure. Note: if a critical section was started once or more, it only takes a single call of this function to end all critical sections.

Definition at line 242 of file process_Management.c.

8.44.2.6 void Sys_Execute_All_EventHandler ( ) [inline]

This function executes all event handlers and processes stored event data

This function executes all event handlers and processes stored event data. First it checks the list of occurred events and then it executes all event handlers of these events

Definition at line 507 of file process_Management.c.

8.44.2.7 void Sys_Execute_Events_in_ProcessList (uint16 *eventID,* sys_pcb_list_element ∗ *elements*) [inline]

This function executes all event handlers and processes stored event data

This function executes all event handlers and processes stored event data. First it checks the list of occurred events and then it executes all event handlers of these events

Parameters

in	*eventID*	event identifier
in	*elements*	list of processes

Definition at line 480 of file process_Management.c.

8.44.2.8 unsigned short Sys_Get_Number_Processes (void) [inline]

This function counts the number of process

This function counts the number of process

Definition at line 65 of file process_Management.c.

8.44.2.9 void Sys_Init_Process_Management (void) [inline]

This function initialises the process management

This function initialises the process management and creates the first elements in the linked list

Definition at line 55 of file process_Management.c.

8.44.2.10 void Sys_Interprocess_EventHandling ( )

This function starts the execution of the event handler and resets the execution time of the process

This function starts the execution of the event handler and resets the execution time of the process

Definition at line 531 of file process_Management.c.

8.44.2.11 void Sys_Kill_Process (uint16 *pid*) [inline]

This function kills a process

This function deletes the syss_process_control_block element and stops a process

Parameters

in	*pid*	This argument is the process identifier

Definition at line 103 of file process_Management.c.

8.44.2.12 void Sys_Kill_Zombies (void) [inline]

This function kills all zombie process

This function deletes all proccesses which are marked as zombies.

Definition at line 180 of file process_Management.c.

8.44.2.13 void Sys_Remove_All_Event_Subscriptions (uint16 *eventID*)

This function removes all subscriptions of any process to event (eventID)

This function removes all subscriptions of any process to event (eventID)

•

Parameters

in	*eventID*	Identifier of the event that has to be removed

Definition at line 547 of file process_Management.c.

8.44.2.14 void Sys_Remove_Event_Subscription (uint16 *pid*, uint16 *eventID*, pEventHandlerFunction *func*)

This function removes subscribed handler function for process (pid) to event (eventID)

This function removes subscribed handler function for process (pid) to event (eventID)

Parameters

in	*pid*	Identifier of the process
in	*eventID*	Identifier of the event that has to be removed
in	*func*	pointer to the subscribed handler function

Definition at line 565 of file process_Management.c.

8.44.2.15 void Sys_Set_Running_Process_to_Zombie ( )

This function puts the running process in the zombie list and switches content to the next ready process

This function puts the running process in the zombie list and switches content to the next ready process

Definition at line 137 of file process_Management.c.

8.44.2.16 void Sys_Start_CriticalSection (void) `[inline]`

Starts a critical section

This Function starts a critical section to prevent the task-scheduling during its exectution

Definition at line 232 of file process_Management.c.

8.44.2.17 bool Sys_Start_Process (pFunction *function*) `[inline]`

This function creates a new sys_process_control_block and add all needed info

This function creates a new sys_process_control_block (in a sys_process_control_block_list_element) which contains all information wich is used to execute this process.

Parameters

in	*function*	This argument points to a function in memory which should be executed as an new task

Definition at line 92 of file process_Management.c.

8.44.2.18 void Sys_Switch_Process (uint16 *pid*)

This function loads all values into the registers of a process with the PID

This function loads all values into the registers of a process with the PID

Parameters

in	*pid*	process id

Definition at line 199 of file process_Management.c.

8.44.2.19 void Sys_Switch_to_next_Process (void)

This function loads all values into the registers of the process which is next in the list.

This function loads all values into the registers of the process which is next in the list.

Definition at line 218 of file process_Management.c.

8.44.2.20 sys_event_data∗ Sys_Wait_For_Condition (uint16 *eventID,* pConditionFunction *function*)

This function blocks the current process.

This function blocks the current process while waiting for an event that sends data which meet the condition.

Parameters

in	*eventID*	Identifier of the event that need to occur
in	*function*	Pointer to the function that represents the condition function (return true if condition is met and continues the process). If function = 0 .. condition is always met.

Returns

sys_event_data ∗ Pointer to the event data struct that contains the values carried by the event

Definition at line 589 of file process_Management.c.

8.44.2.21 sys_event_data∗ Sys_Wait_For_Event (uint16 *eventID*) [inline]

This function blocks the current process.

This function blocks the current process and waits for the occurrence of event (eventID).

Parameters

eventID	ID of the event

Returns

sys_event_data∗ Pointer to the event data struct that contains the values carried by the event

Definition at line 626 of file process_Management.c.

8.44.2.22 void Sys_Yield (void)

This function blocks the current process.

This function blocks the current process and let the next process be executed.

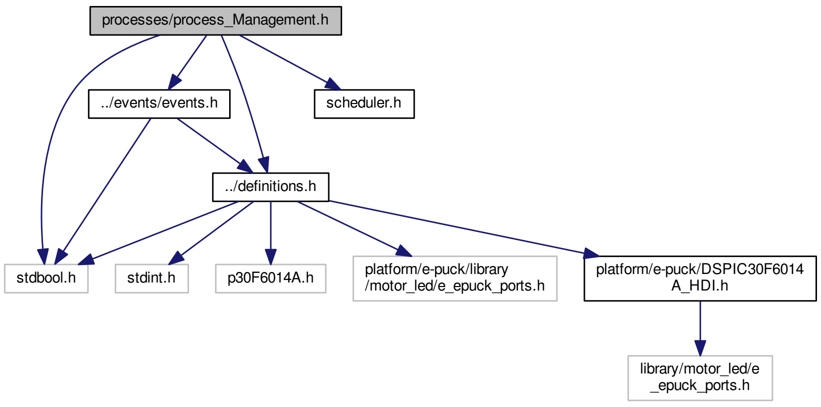
Definition at line 636 of file process_Management.c.

8.45 processes/process_Management.h File Reference

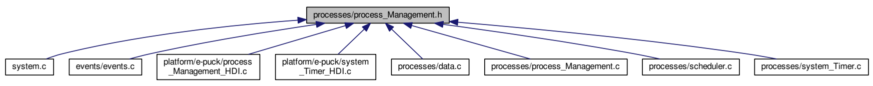
This file includes all functions wich are needed to manage processes (e.g. task creation, switching, termination)

```
#include <stdbool.h>
#include "../definitions.h"
#include "../events/events.h"
#include "scheduler.h"
```

Include dependency graph for process_Management.h:


processes/process_Management.h
../events/events.h
scheduler.h
../definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014 A_HDI.h
library/motor_led/e _epuck_ports.h

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


processes/process_Management.h
system.c
events/events.c
platform/e-puck/process _Management_HDI.c
platform/e-puck/system _Timer_HDI.c
processes/data.c
processes/process_Management.c
processes/scheduler.c
processes/system_Timer.c

Macros

- #define DEFAULT_PROCESS_STACK_SIZE 200

Functions

- void Sys_Switch_Process (uint16 pid)
- void Sys_Switch_to_next_Process (void)
- bool Sys_Start_Process (pFunction function)
- void Sys_Kill_Process (uint16 pid)
- void Sys_Kill_Zombies (void)
- void Sys_Yield (void)
- void Sys_Init_Process_Management (void)
- unsigned short Sys_Get_Number_Processes (void)
- void Sys_Start_CriticalSection (void)
- void Sys_End_CriticalSection (void)
- bool Sys_Add_Event_Subscription (uint16 pid, uint16 eventID, pEventHandlerFunction func, pCondition↩ Function cond)
- void Sys_Remove_Event_Subscription (uint16 pid, uint16 eventID, pEventHandlerFunction func)
- void Sys_Remove_All_Event_Subscriptions (uint16 eventID)
- void Sys_Add_Event_to_Process (uint16 pid, uint16 eventID, void ∗data, uint16 length)
- void Sys_Execute_All_EventHandler ()
- void Sys_Clear_EventData (sys_event_data ∗∗data)

- sys_event_data ∗ Sys_Wait_For_Event (uint16 eventID)
- sys_event_data ∗ Sys_Wait_For_Condition (uint16 eventID, pConditionFunction function)

8.45.1 Detailed Description

This file includes all functions wich are needed to manage processes (e.g. task creation, switching, termination)

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

08 July 2014

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.45.2 Macro Definition Documentation

8.45.2.1 #define DEFAULT_PROCESS_STACK_SIZE 200

process default stack size

Definition at line 110 of file process_Management.h.

8.45.3 Function Documentation

8.45.3.1 bool Sys_Add_Event_Subscription (uint16 *pid,* uint16 *eventID,* pEventHandlerFunction *func,* pConditionFunction *cond*)

Puts a process on the blocked list and stops its execution (if it's executed)

Puts a process on the blocked list and stops its execution (if it's executed)

Parameters

in	*pid*	Process ID
in	*eventID*	The Id of the event which can put the process (PID) back on the ready list
in	*func*	The function that handles the event
in	*cond*	The condition under which the handler is executed

Returns

bool Was the event-handler successfully added?

Definition at line 336 of file process_Management.c.

8.45.3.2 void Sys_Add_Event_to_Process (uint16 *pid,* uint16 *eventID,* void ∗ *data,* uint16 *length*)

This function adds the event-data to the local list of the process (pid).

This function adds the event-data to the local list of the process (pid).

Parameters

`in`	*pid*	process identifier
`in`	*eventID*	event identifier
`in`	*data*	memory that contains the value of the occurred event
`in`	*length*	length of the data (bytes)

Definition at line 390 of file process_Management.c.

8.45.3.3 void Sys_Clear_EventData (sys_event_data ** *data*) `[inline]`

This function removes and frees a list of sys_event_data

This function removes and frees a list of sys_event_data

Parameters

`in,out`	*data*	pointer to the event_data (list)

Returns

void

Definition at line 219 of file data.c.

8.45.3.4 void Sys_End_CriticalSection (void) `[inline]`

Ends a critical section

This Function ends a critical section. The task-scheduling can now occure. Note: if a critical section was started once or more, it only takes a single call of this function to end all critical sections.

Definition at line 242 of file process_Management.c.

8.45.3.5 void Sys_Execute_All_EventHandler ( ) `[inline]`

This function executes all event handlers and processes stored event data

This function executes all event handlers and processes stored event data. First it checks the list of occurred events and then it executes all event handlers of these events

Definition at line 507 of file process_Management.c.

8.45.3.6 unsigned short Sys_Get_Number_Processes (void) `[inline]`

This function counts the number of process

This function counts the number of process

Definition at line 65 of file process_Management.c.

8.45.3.7 void Sys_Init_Process_Management (void) `[inline]`

This function initialises the process management

This function initialises the process management and creates the first elements in the linked list

Definition at line 55 of file process_Management.c.

8.45.3.8 void Sys_Kill_Process (uint16 *pid*) [inline]

This function kills a process

This function deletes the syss_process_control_block element and stops a process

Parameters

in	*pid*	This argument is the process identifier

Definition at line 103 of file process_Management.c.

8.45.3.9 void Sys_Kill_Zombies (void) [inline]

This function kills all zombie process

This function deletes all proccesses which are marked as zombies.

Definition at line 180 of file process_Management.c.

8.45.3.10 void Sys_Remove_All_Event_Subscriptions (uint16 *eventID*)

This function removes all subscriptions of any process to event (eventID)

This function removes all subscriptions of any process to event (eventID)

- **Parameters**

in	*eventID*	Identifier of the event that has to be removed

Definition at line 547 of file process_Management.c.

8.45.3.11 void Sys_Remove_Event_Subscription (uint16 *pid,* uint16 *eventID,* pEventHandlerFunction *func*)

This function removes subscribed handler function for process (pid) to event (eventID)

This function removes subscribed handler function for process (pid) to event (eventID)

Parameters

in	*pid*	Identifier of the process
in	*eventID*	Identifier of the event that has to be removed
in	*func*	pointer to the subscribed handler function

Definition at line 565 of file process_Management.c.

8.45.3.12 void Sys_Start_CriticalSection (void) [inline]

Starts a critical section

This Function starts a critical section to prevent the task-scheduling during its exectution

Definition at line 232 of file process_Management.c.

8.45.3.13 bool Sys_Start_Process (pFunction *function*) [inline]

This function creates a new sys_process_control_block and add all needed info

This function creates a new sys_process_control_block (in a sys_process_control_block_list_element) which contains all information wich is used to execute this process.

Parameters

`in`	*function*	This argument points to a function in memory which should be executed as an new task

Definition at line 92 of file process_Management.c.

8.45.3.14 void Sys_Switch_Process (uint16 *pid*)

This function loads all values into the registers of a process with the PID

This function loads all values into the registers of a process with the PID

Parameters

`in`	*pid*	process id

Definition at line 199 of file process_Management.c.

8.45.3.15 void Sys_Switch_to_next_Process (void)

This function loads all values into the registers of the process which is next in the list.

This function loads all values into the registers of the process which is next in the list.

Definition at line 218 of file process_Management.c.

8.45.3.16 sys_event_data∗ Sys_Wait_For_Condition (uint16 *eventID*, pConditionFunction *function*)

This function blocks the current process.

This function blocks the current process while waiting for an event that sends data which meet the condition.

Parameters

`in`	*eventID*	Identifier of the event that need to occur
`in`	*function*	Pointer to the function that represents the condition function (return true if condition is met and continues the process). If function = 0 .. condition is always met.

Returns

sys_event_data ∗ Pointer to the event data struct that contains the values carried by the event

Definition at line 589 of file process_Management.c.

8.45.3.17 sys_event_data∗ Sys_Wait_For_Event (uint16 *eventID*) `[inline]`

This function blocks the current process.

This function blocks the current process and waits for the occurrence of event (eventID).

Parameters

eventID	ID of the event

Returns

sys_event_data∗ Pointer to the event data struct that contains the values carried by the event

Definition at line 626 of file process_Management.c.

8.45.3.18 void Sys_Yield (void)

This function blocks the current process.

This function blocks the current process and let the next process be executed.

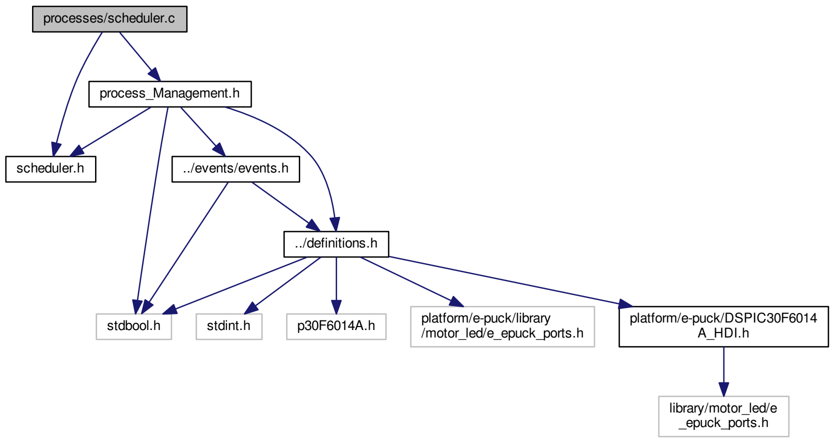
Definition at line 636 of file process_Management.c.

8.46 processes/scheduler.c File Reference

This file includes all functions wich are needed to specify a scheduling algorithm.

```
#include "scheduler.h"
#include "process_Management.h"
```

Include dependency graph for scheduler.c:



Functions

- void Sys_Scheduler_RoundRobin (void)
- void Sys_Set_Defaults_Info (sys_scheduler_info ∗sct)

8.46.1 Detailed Description

This file includes all functions wich are needed to specify a scheduling algorithm.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

{07 July 2014}

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.46.2 Function Documentation

8.46.2.1 void Sys_Scheduler_RoundRobin (void)

This function represents the Schedling algorithm

This function shows the implementation of the RoundRobin Scheduling algorithm

Definition at line 25 of file scheduler.c.

8.46.2.2 void Sys_Set_Defaults_Info (**sys_scheduler_info** ∗ *sct*) `[inline]`

This function sets default values to the scheduling struct

This function sets the default values in a sys_scheduler_info struct

Parameters

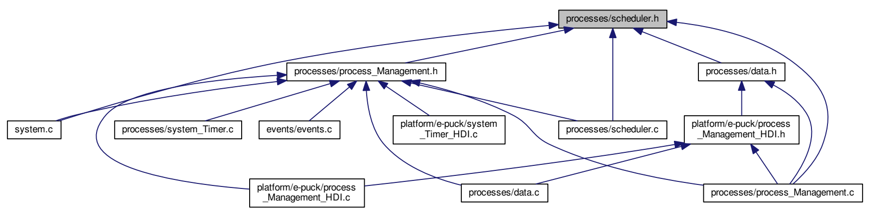
`in,out`	*sct*	This is a pointer to a sys_scheduler_info struct

Definition at line 47 of file scheduler.c.

8.47 processes/scheduler.h File Reference

This file includes all functions wich are needed to specify a scheduling algorithm.

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


processes/scheduler.h
processes/process_Management.h
processes/data.h
system.c
processes/system_Timer.c
events/events.c
platform/e-puck/system _Timer_HDI.c
processes/scheduler.c
platform/e-puck/process _Management_HDI.h
platform/e-puck/process _Management_HDI.c
processes/data.c
processes/process_Management.c

Data Structures

- struct sys_scheduler_info_s

  The scheduling information for each process.

Macros

- #define SYS_PROCESS_STATE_BABY 0xBABE
- #define SYS_PROCESS_STATE_RUNNING 0xFFFF
- #define SYS_PROCESS_STATE_BLOCKED 0xBCED
- #define SYS_PROCESS_STATE_WAITING 0x5555
- #define SYS_PROCESS_STATE_ZOMBIE 0xDEAD
- #define SYS_PROCESS_PRIORITY_SYSTEM 0xFFFF
- #define SYS_PROCESS_PRIORITY_HIGH 0x0FFF

- #define SYS_PROCESS_PRIORITY_NORMAL 0x00FF
- #define SYS_PROCESS_PRIORITY_LOW 0x000F

Typedefs

- typedef struct sys_scheduler_info_s sys_scheduler_info

  The scheduling information for each process.

Functions

- void Sys_Scheduler_RoundRobin (void)
- void Sys_Set_Defaults_Info (sys_scheduler_info ∗sct)

8.47.1 Detailed Description

This file includes all functions wich are needed to specify a scheduling algorithm.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

{07 July 2014}

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.47.2 Macro Definition Documentation

8.47.2.1 #define SYS_PROCESS_PRIORITY_HIGH 0x0FFF

Definition at line 29 of file scheduler.h.

8.47.2.2 #define SYS_PROCESS_PRIORITY_LOW 0x000F

Definition at line 31 of file scheduler.h.

8.47.2.3 #define SYS_PROCESS_PRIORITY_NORMAL 0x00FF

Definition at line 30 of file scheduler.h.

8.47.2.4 #define SYS_PROCESS_PRIORITY_SYSTEM 0xFFFF

process priority values Process priority: System = highest

Definition at line 28 of file scheduler.h.

8.47.2.5 #define SYS_PROCESS_STATE_BABY 0xBABE

process state values State to indicate that a process is created but not yet ready to be executed

Definition at line 21 of file scheduler.h.

8.47.2.6 #define SYS_PROCESS_STATE_BLOCKED 0xBCED

State to indicate that a process is blocked and waits till an event occurs

Definition at line 23 of file scheduler.h.

8.47.2.7 #define SYS_PROCESS_STATE_RUNNING 0xFFFF

State to indicate that a process is executed

Definition at line 22 of file scheduler.h.

8.47.2.8 #define SYS_PROCESS_STATE_WAITING 0x5555

State to indicate that a process is waiting to be executed

Definition at line 24 of file scheduler.h.

8.47.2.9 #define SYS_PROCESS_STATE_ZOMBIE 0xDEAD

State to indicate that a process is about to be deleted

Definition at line 25 of file scheduler.h.

8.47.3 Typedef Documentation

8.47.3.1 typedef struct sys_scheduler_info_s sys_scheduler_info

The scheduling information for each process.

This struct defines all values wich are needed for the scheduling algorithm

8.47.4 Function Documentation

8.47.4.1 void Sys_Scheduler_RoundRobin (void)

This function represents the Schedling algorithm

This function shows the implementation of the RoundRobin Scheduling algorithm

Definition at line 25 of file scheduler.c.

8.47.4.2 void Sys_Set_Defaults_Info (sys_scheduler_info ∗ *sct*) [inline]

This function sets default values to the scheduling struct

This function sets the default values in a sys_scheduler_info struct

Parameters

in,out	*sct*	This is a pointer to a sys_scheduler_info struct

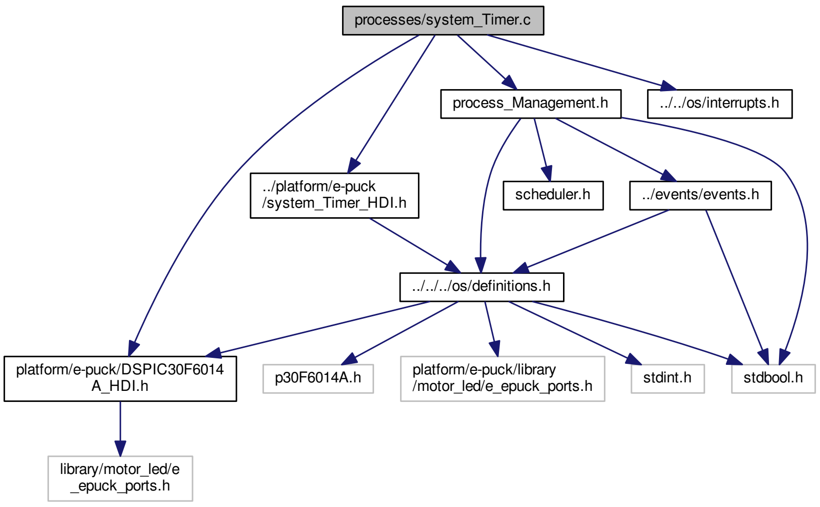
Definition at line 47 of file scheduler.c.

8.48 processes/system_Timer.c File Reference

This file includes all hardware dependent functions, which are nesessary to initialise, configure and run the system Time.

```
#include "../platform/e-puck/system_Timer_HDI.h"
#include "process_Management.h"
#include "../platform/e-puck/DSPIC30F6014A_HDI.h"
#include "../../os/interrupts.h"
```

Include dependency graph for system_Timer.c:


processes/system_Timer.c
process_Management.h
../../os/interrupts.h
../platform/e-puck /system_Timer_HDI.h
scheduler.h
../events/events.h
../../../os/definitions.h
platform/e-puck/DSPIC30F6014 A_HDI.h
p30F6014A.h
platform/e-puck/library /motor_led/e_epuck_ports.h
stdint.h
stdbool.h
library/motor_led/e _epuck_ports.h

Functions

- void Sys_todo_SystemTimer ()
- void Sys_Init_SystemTimer (pFunction scheduler)
- void Sys_Start_SystemTimer ()
- void Sys_Stop_SystemTimer ()
- void Sys_Continue_SystemTimer ()
- void Sys_Reset_SystemTimer ()
- void Sys_Disable_TimerInterrupt (void)
- void Sys_Enable_TimerInterrupt (void)
- void Sys_Force_TimerInterrupt (void)

8.48.1 Detailed Description

This file includes all hardware dependent functions, which are nesessary to initialise, configure and run the system Time.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

{07 July 2014}

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.48.2 Function Documentation

8.48.2.1 void Sys_Continue_SystemTimer ( ) `[inline]`

Deactivates the Timer1 Interrupt

This Function deactivated the Timer1 Interrupt

Definition at line 59 of file system_Timer.c.

8.48.2.2 void Sys_Disable_TimerInterrupt (void) `[inline]`

Disables the Timer1 interrupt

Disables the Timer1 interrupt and sets the interrupt flag to 0

Definition at line 101 of file system_Timer.c.

8.48.2.3 void Sys_Enable_TimerInterrupt (void) `[inline]`

Enables the Timer1 interrupt

Enables the Timer1 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 111 of file system_Timer.c.

8.48.2.4 void Sys_Force_TimerInterrupt (void) `[inline]`

Enables the Timer1 interrupt

Enables the Timer1 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 121 of file system_Timer.c.

8.48.2.5 void Sys_Init_SystemTimer (**pFunction** *scheduler*) `[inline]`

Function to initialise the system timer

This Function sets the Timer0 of the DSPIC 30F6014A for timer interfvals of 10 ms. The timer will be startet with Start_SystemTimer_HDI()

Parameters

`in,out`	*scheduler*	This is a pointer t an callback function, which schuld becalled whenever a timer interrupt is emmitted.

Definition at line 28 of file system_Timer.c.

8.48.2.6 void Sys_Reset_SystemTimer ( ) `[inline]`

Resets the Timer1 value to the initial value

This Function resets the Timer1 value

Definition at line 69 of file system_Timer.c.

8.48.2.7 void Sys_Start_SystemTimer (void) `[inline]`

Function to starts the initialised system timer

This Function starts the Timer0 of the DSPIC 30F6014A for timer interfvals of 10 ms. The MUST be initialised first with Init_SystemTimer_HDI()

Definition at line 39 of file system_Timer.c.

8.48.2.8 void Sys_Stop_SystemTimer ( ) `[inline]`

Activates the Timer1 Interrupt

This Function activated the Timer1 Interrupt

Definition at line 49 of file system_Timer.c.

8.48.2.9 void Sys_todo_SystemTimer ( ) `[inline]`

This function is executed periodically by the system timer interrupt

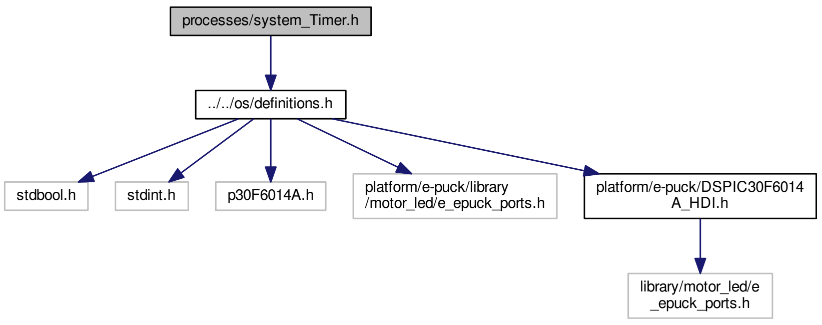
This function is executed periodically by the system timer interrupt. It kills all zombies, executes event handlers and executes the scheduling algorithm.

Definition at line 79 of file system_Timer.c.

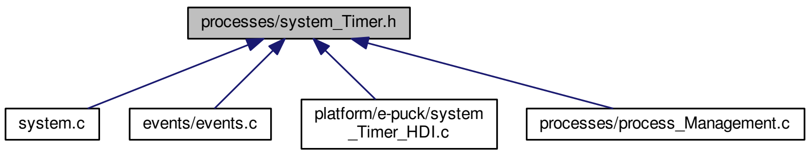
8.49 processes/system_Timer.h File Reference

This file includes all hardware dependent functions, which are nesessary to initialise, configure and run the system Time.

`#include "../../os/definitions.h"`
Include dependency graph for system_Timer.h:



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



Functions

- void Sys_Init_SystemTimer (pFunction)
- void Sys_Start_SystemTimer (void)
- void Sys_Stop_SystemTimer ()
- void Sys_Continue_SystemTimer ()
- void Sys_Disable_TimerInterrupt (void)
- void Sys_Enable_TimerInterrupt (void)
- void Sys_Force_TimerInterrupt (void)
- void Sys_Reset_SystemTimer ()
- void Sys_todo_SystemTimer ()

8.49.1 Detailed Description

This file includes all hardware dependent functions, which are nesessary to initialise, configure and run the system Time.

Author

Stefan M. Trenkwalder `s.trenkwalder@openswarm.org`

Version

1.0

Date

{07 July 2014}

Copyright

adapted FreeBSD License (see `http://openswarm.org/license`)

8.49.2 Function Documentation

8.49.2.1 void Sys_Continue_SystemTimer ( ) `[inline]`

Deactivates the Timer1 Interrupt

This Function deactivated the Timer1 Interrupt

Definition at line 59 of file system_Timer.c.

8.49.2.2 void Sys_Disable_TimerInterrupt (void) `[inline]`

Disables the Timer1 interrupt

Disables the Timer1 interrupt and sets the interrupt flag to 0

Definition at line 101 of file system_Timer.c.

8.49.2.3 void Sys_Enable_TimerInterrupt (void) `[inline]`

Enables the Timer1 interrupt

Enables the Timer1 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 111 of file system_Timer.c.

8.49.2.4 void Sys_Force_TimerInterrupt (void) `[inline]`

Enables the Timer1 interrupt

Enables the Timer1 interrupt and leaves the interrupt flag to its value. If the flag was set, the Timer1 interrupt will be emitted after executing this function.

Definition at line 121 of file system_Timer.c.

8.49.2.5 void Sys_Init_SystemTimer (pFunction *scheduler*) `[inline]`

Function to initialise the system timer

This Function sets the Timer0 of the DSPIC 30F6014A for timer interfvals of 10 ms. The timer will be startet with Start_SystemTimer_HDI()

Parameters

`in,out`	*scheduler*	This is a pointer t an callback function, which schuld becalled whenever a timer interrupt is emmitted.

Definition at line 28 of file system_Timer.c.

8.49.2.6 void Sys_Reset_SystemTimer ( ) [inline]

Resets the Timer1 value to the initial value

This Function resets the Timer1 value

Definition at line 69 of file system_Timer.c.

8.49.2.7 void Sys_Start_SystemTimer (void) [inline]

Function to starts the initialised system timer

This Function starts the Timer0 of the DSPIC 30F6014A for timer interfvals of 10 ms. The MUST be initialised first with Init_SystemTimer_HDI()

Definition at line 39 of file system_Timer.c.

8.49.2.8 void Sys_Stop_SystemTimer ( ) [inline]

Activates the Timer1 Interrupt

This Function activated the Timer1 Interrupt

Definition at line 49 of file system_Timer.c.

8.49.2.9 void Sys_todo_SystemTimer ( ) [inline]

This function is executed periodically by the system timer interrupt

This function is executed periodically by the system timer interrupt. It kills all zombies, executes event handlers and executes the scheduling algorithm.

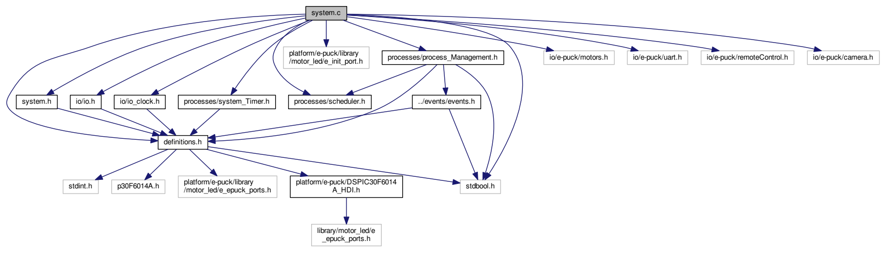
Definition at line 79 of file system_Timer.c.

8.50 system.c File Reference

includes system calls that initialise and configure the operating system.

```
#include "definitions.h"
#include "system.h"
#include "processes/system_Timer.h"
#include "processes/scheduler.h"
#include "processes/process_Management.h"
#include "platform/e-puck/library/motor_led/e_init_port.h"
#include "io/io.h"
#include "io/io_clock.h"
#include "io/e-puck/motors.h"
#include "io/e-puck/uart.h"
#include "io/e-puck/remoteControl.h"
#include "io/e-puck/camera.h"
```

Include dependency graph for system.c:



Functions

- void Sys_Init_Kernel ()
- void Sys_Start_Kernel (void)

8.50.1 Detailed Description

includes system calls that initialise and configure the operating system.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

2015

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

In short, Openswarm can be executed as shown in the following example

```
#include "os/system.h"

int main(void){
     //initialise some global or local variables

    Sys_Init_Kernel();

     //do some preperation before executing OpenSwarm and user applications

     Sys_Start_Kernel();
    while(1){
        //do nothing
    }
}
```

8.50.2 Function Documentation

8.50.2.1 void Sys_Init_Kernel (void)

Function to initialise the hardware

This Function sets the system Timer (Timer0) and sets an scheduling algorithm. It also intitalises I/O devices (e.g. if e-puck is used: motor, UART, remote control, and camera)

Postcondition

To start OpenSwarm, Sys_Start_Kernel() mast be executed after the initialisation.

Remarks

Code can be executed between initialisation and start of the kernel. But, note that you can only execute code that does not depend on an active OpenSwarm.

Definition at line 64 of file system.c.

8.50.2.2 void Sys_Start_Kernel (void)

Function to start the the system timer

This Function starts all functions of the operating system. The system MUST HAVE BEEN INITIALISED before.

Precondition

System must be initialised with Sys_Init_Kernel().

Remarks

Code can be executed between initialisation and start of the kernel. But, note that you can only execute code that does not depend on an active OpenSwarm.

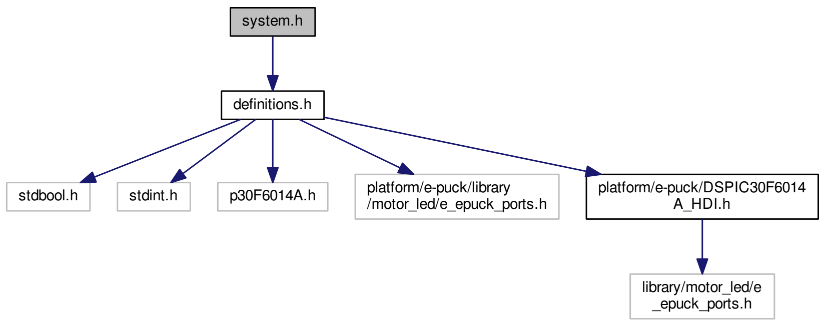
Definition at line 104 of file system.c.

8.51 system.h File Reference

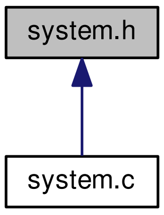
initiaises and starts OpenSwarm.

`#include "definitions.h"`

Include dependency graph for system.h:


system.h
definitions.h
stdbool.h
stdint.h
p30F6014A.h
platform/e-puck/library/motor_led/e_epuck_ports.h
platform/e-puck/DSPIC30F6014A_HDI.h
library/motor_led/e_epuck_ports.h

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



Macros

- #define SYS_MOTOR_USED
- #define SYS_UART1_USED
- #define SYS_REMOTECONTROL_USED
- #define SYS_CAMERA_USED

Functions

- void Sys_Init_Kernel (void)
- void Sys_Start_Kernel (void)

8.51.1 Detailed Description

initiaises and starts OpenSwarm.

Author

Stefan M. Trenkwalder s.trenkwalder@openswarm.org

Version

1.0

Date

{07 July 2014}

Copyright

adapted FreeBSD License (see http://openswarm.org/license)

8.51.2 Macro Definition Documentation

8.51.2.1 #define SYS_CAMERA_USED

Define this preprocessor symbol to use the camera

Definition at line 84 of file system.h.

8.51.2.2 #define SYS_MOTOR_USED

Define this preprocessor symbol to use motors

Definition at line 81 of file system.h.

8.51.2.3 #define SYS_REMOTECONTROL_USED

Define this preprocessor symbol to receive remote control signals

Definition at line 83 of file system.h.

8.51.2.4 #define SYS_UART1_USED

Define this preprocessor symbol to use UART1

Definition at line 82 of file system.h.

8.51.3 Function Documentation

8.51.3.1 void Sys_Init_Kernel (void)

Function to initialise the hardware

This Function sets the system Timer (Timer0) and sets an scheduling algorithm. It also intitalises I/O devices (e.g. if e-puck is used: motor, UART, remote control, and camera)

Postcondition

To start OpenSwarm, Sys_Start_Kernel() mast be executed after the initialisation.

Remarks

Code can be executed between initialisation and start of the kernel. But, note that you can only execute code that does not depend on an active OpenSwarm.

Definition at line 64 of file system.c.

8.51.3.2 void Sys_Start_Kernel (void)

Function to start the the system timer

This Function starts all functions of the operating system. The system MUST HAVE BEEN INITIALISED before.

Precondition

System must be initialised with Sys_Init_Kernel().

Remarks

Code can be executed between initialisation and start of the kernel. But, note that you can only execute code that does not depend on an active OpenSwarm.

Definition at line 104 of file system.c.

Index