Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

SENSOR TERMINAL BOARD by dresden elektronik
Reference Manual

Generated by Doxygen 1.5.1-p1

Mon Nov 5 13:22:14 2007

Contents

Chapter 1

SENSOR TERMINAL BOARD by dresden elektronik Main Page

1.1 Introduction

This CDROM contains all necessary tools for developing, building and debugging software for the SensTermBoard on the Windows (TM) plattform. All **examples** (p. 3) are tested with the listed and delivered software versions.

1.2 Copyright

The different included software packages are subject to different licences. See its detailed conditions while installing an application. Every included application was choosen instead of others because its special licence allows the use and redistribution for personal and commercial purposes. The included examples are also open source as you can read on the **copyright page** (p. 111).

1.3 Installation

1.3.1 Applications

For installation of the required applications you can start the software installers directly from this documentation:

- `AVR-Studio version 4.13 B557`: Free IDE from ATMEL
- `WinAVR version 20070525`: gnu compiler collection tool chain, gcc version 4.1.1

- `rdk230wpan version 1.3`: free MAC layer software stack from ATMEL for 802.15.4 mesh networks for AT89RF230 radio and ATMEGA1281 controller

1.3.2 Attention!

The electrical circuit of the SensTermBoard does not allow the parallel use of the batteries and the USB/DC connection. Please make sure that the batteries are taken out of the RCB when an external power source is connected.

1.3.3 USB-Driver

When you plug the PC's USB cable into the SensTermBoard, Windows (TM) asks you for the location of the USB driver for the board. It can be found in `".\3dParty\FTDI\CDM 2.00.00"` on the CDROM. There is also a newer version (CDM 2.02.04 WHQL Certified), but this one has a serious bug at the API layer and is not recommended. FTDI is working on a new version.

1.3.4 Building The Documentation

This documentation was generated by the use of the following tools

- `Doxygen 1.5.1 p1`: free source documentation system from Dimitri van Heesch
- `graphviz-2.12.exe`: free generator for tree graphs from AT&T Research Labs

which are called in the batch file

```
.\Doc\Doxygen\doxygen.bat
```

1.3.5 Documentation As PDF

The related manual `refman.pdf` was generated out of the doxygen output by using the tools

- `LaTeX`
- `GhostScript`

1.4 Getting Startet

For a short guide how to build and debug applications on the SensTermBoard see the page **Getting Startet** (p. 113). There you can find a step by step guide from powering up to debugging.

1.5 Examples

On the CDROM are a lot of miniature example applications which are demonstrating a special part of the SensTermBoard. You can find its documentation on the `files page`.

1.6 CDROM Directory Structure

```
+--- 3dParty              Applications, Sources, Drivers
|   +--- ATMEL            IDE, RF Sources
|   |   +--- AVR-Studio   IDE
|   |   +--- rdk230wpan   RF Sources
|   +--- Doxygen          Source documentation
|   +--- FTDI             USB-Chip Driver
|   +--- WinAVR           Open Source ANSI-C compiler tool chain
+--- Doc                  Documentation around the SensTermBoard
|   +--- DataSheets       Data sheets for  ATMEGA1281,  AT86RF230
|   +--- Doxygen          this documentation and its config files
|   +--- Images           Images used for this documentation
|   +--- PCB              PCB files (schematic, layout) for SensTermBoard
(p. 114) and RCB (p. 115)
+--- Examples             examples for the the board's components
    +---Button (p. 18)          demonstrates init and use of the button
    +---LEDLoop (p. 25)         Blinking LED under use of a delay loop
    +---LEDTimer (p. 29)        Blinking LED under use of timer 4
    +---Temperature (p. 34)     Using the thermistor for temperature measurement
    +---TWIEeprom (p. 44)       Using the I2C / TWI modul for access to an external eeprom
    +---Uart (p. 61)            "Hello World" with 9600 Baud over the UART0
    +---Usb (p. 66)             "Hello World" over the USB chip FT245
    +---XRAM (p. 71)            How to use the external 32k (XRAM)
    +--- Z-LinkWireless   Devices (p. 97) notify a Coordinator
(p. 77) via RF about its temperatures, which it sends over USB to a PC
```

Chapter 2

SENSOR TERMINAL BOARD by dresden elektronik Data Structure Index

2.1 SENSOR TERMINAL BOARD by dresden elektronik Data Structures

Here are the data structures with brief descriptions:

Chapter 3

SENSOR TERMINAL BOARD by dresden elektronik File Index

3.1 SENSOR TERMINAL BOARD by dresden elektronik File List

Here is a list of all files with brief descriptions:

Chapter 4

SENSOR TERMINAL BOARD by dresden elektronik Page Index

4.1 SENSOR TERMINAL BOARD by dresden elektronik Related Pages

Here is a list of all related documentation pages:

Chapter 5

SENSOR TERMINAL BOARD by dresden elektronik Data Structure Documentation

5.1 association_entry_t Struct Reference

Data Fields

- bool **associated**
- uint64_t **long_addr**
- uint16_t **temperature**

5.1.1 Detailed Description

Definition at line 96 of file coord.c.

5.1.2 Field Documentation

5.1.2.1 bool association_entry_t::associated

Definition at line 98 of file coord.c.

Referenced by usr_mcps_data_ind(), and usr_mlme_comm_status_ind().

5.1.2.2 uint64_t association_entry_t::long_addr

Definition at line 99 of file coord.c.

Referenced by association_table_init(), and mac_register_device().

5.1.2.3 uint16_t association_entry_t::temperature

Definition at line 100 of file coord.c.

Referenced by usr_mcps_data_ind().

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

- Examples/Z-LinkWireless/**coord.c**

5.2 coord_status_t Struct Reference

Data Fields

- uint8_t **handle**
- uint8_t **led_value**
- **coord_state_t state**

5.2.1 Detailed Description

Definition at line 114 of file coord.c.

5.2.2 Field Documentation

5.2.2.1 uint8_t coord_status_t::handle

Definition at line 116 of file coord.c.

Referenced by application_init(), and mac_send_data().

5.2.2.2 uint8_t coord_status_t::led_value

Definition at line 117 of file coord.c.

Referenced by application_init(), and mac_send_data().

5.2.2.3 coord_state_t coord_status_t::state

Definition at line 118 of file coord.c.

Referenced by usr_mlme_associate_ind(), usr_mlme_reset_conf(), usr_mlme_scan_conf(), usr_mlme_set_conf(), and usr_mlme_start_conf().

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

- Examples/Z-LinkWireless/**coord.c**

5.3 device_status_t Struct Reference

Data Fields

- bool **led**
- bool **switch_pressed**
- uint16_t **device_short_address**
- uint8_t **coord_address_mode**
- uint64_t **coord_address**
- uint16_t **pan_id**
- uint8_t **logical_channel**
- uint8_t **msdu_handle**
- **device_state_t state**

5.3.1 Detailed Description

Definition at line 86 of file device.c.

5.3.2 Field Documentation

5.3.2.1 bool device_status_t::led

Definition at line 88 of file device.c.

5.3.2.2 bool device_status_t::switch_pressed

Definition at line 89 of file device.c.

5.3.2.3 uint16_t device_status_t::device_short_address

Definition at line 90 of file device.c.

Referenced by switch_task(), and usr_mlme_associate_conf().

5.3.2.4 uint8_t device_status_t::coord_address_mode

Definition at line 91 of file device.c.

Referenced by mac_associate(), switch_task(), and usr_mlme_scan_conf().

5.3.2.5 uint64_t device_status_t::coord_address

Definition at line 92 of file device.c.

Referenced by mac_associate(), switch_task(), and usr_mlme_scan_conf().

5.3.2.6 uint16_t device_status_t::pan_id

Definition at line 93 of file device.c.

Referenced by mac_associate(), switch_task(), usr_mcps_data_ind(), and usr_mlme_scan_conf().

5.3.2.7 uint8_t device_status_t::logical_channel

Definition at line 94 of file device.c.

Referenced by mac_associate(), and usr_mlme_scan_conf().

5.3.2.8 uint8_t device_status_t::msdu_handle

Definition at line 95 of file device.c.

Referenced by switch_task().

5.3.2.9 device_state_t device_status_t::state

Definition at line 96 of file device.c.

Referenced by switch_task(), usr_mcps_data_ind(), usr_mlme_associate_conf(), usr_mlme_reset_conf(), and usr_mlme_scan_conf().

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

- Examples/Z-LinkWireless/**device.c**

Chapter 6

SENSOR TERMINAL BOARD by dresden elektronik File Documentation

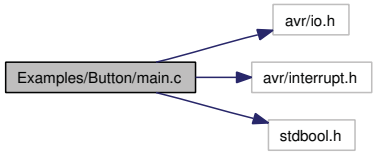
6.1 Doc/Doxygen/SensTermBoard.dox File Reference

6.2 Examples/Button/main.c File Reference

SensTermBoard Example: LED blinks with timer interrupt.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
```

Include dependency graph for main.c:



Defines

- #define **IO_LED_AD** (0x4000)
- #define **IO_BUT_AD** (0x4000)

Functions

- static void **led_init** (void)

  Initializes the led hardware.

- static void **led_set** (int led_nr, bool OnOff)

  Set LED nr's state to on or off.

- static void **button_init** (void)

  Initializes the button hardware.

- static bool **button_pressed** (void)

  Gives the button's state back.

- void **timer_init** (void)

  Init timer 4 for 1 millisecond interrupting.

- **ISR** (TIMER4_COMPB_vect)

  timer 4 capture and compare interrupt routine

- int **main** (void)

  Main function of the demo application.

Variables

- static volatile unsigned char * **pIO_LED** = (unsigned char *) IO_LED_AD
- static volatile unsigned char * **pIO_BUT** = (unsigned char *) IO_BUT_AD
- static uint8_t **LED** = 0x00

6.2.1 Detailed Description

SensTermBoard Example: LED blinks with timer interrupt.

Changes the active LED when the button is pressed. The button state is scaned from the timer 4 interrupt and debaunced through testing more times for the new state

Author:

Definition in file **main.c**.

6.2.2 Define Documentation

6.2.2.1 #define IO_BUT_AD (0x4000)

Definition at line 50 of file main.c.

6.2.2.2 #define IO_LED_AD (0x4000)

Definition at line 49 of file main.c.

6.2.3 Function Documentation

6.2.3.1 static void button_init (void) [static]

Initializes the button hardware.

Initializes the button hardware so that further calls to button_pressed works

Definition at line 103 of file main.c.

Referenced by main().

```
104 {
105   ;
106 }
```

Here is the caller graph for this function:

6.2.3.2 static bool button_pressed (void) [static]

Gives the button's state back.

Returns:

bool false: not pressed, true: pressed

Definition at line 116 of file main.c.

References pIO_BUT.

Referenced by ISR().

```
117 {
118   return ((*pIO_BUT & 0x01) ? true : false);
119 }
```

Here is the caller graph for this function:

6.2.3.3 ISR (TIMER4_COMPB_vect)

timer 4 capture and compare interrupt routine

timer for interrupts every 1 millisecond with a compare event. The compare counter gets resetet and every 10 milliseconds gets the state of the button scanned for changes. On the first push button the active LED is changed and then waited for a 50 milliseconds long release of the button

Definition at line 148 of file main.c.

References button_pressed(), and led_set().

```
149 {
150   static int ms = 0;                              // memory for milliseconds
151   static bool led_0 = true;                       // memory for active led
152   static int  button_cnt = 0;                     // counter for button debounce
153   static bool button_state = false;               // memory for button state
154
155   TCNT4 = 0;                                      // clear timer
156   if (10 <= ++ms)                                 // 10 milliseconds over?
157   {
158     ms = 0;                                       // reset milliseconds counter
159
160     switch (button_state)                         // button is in...
161     {
162       case false:                                 // Off State?
163         if (button_pressed())                     // button pressed?
164         {
165           button_state = true;                    // On State
166           led_set(0, led_0);                      // Set LED 0 to on, if activ, otherwise off
167           led_set(1,!led_0);                      // Set LED 1 to on, if activ, otherwise off
168           led_0 = !led_0;                         // change active LED
169         }
170         break;
171
172       case true:                                  // On State?
173         if (!button_pressed())                    // button released?
174         {
175           if (5 <= ++button_cnt)                  // for a minimum time of 50 milliseconds?
176           {
177             button_state = false;
178           }
179         }
180         else                                      // button bounced? reset counter
181         {
182           button_cnt = 0;
183         }
184         break;
185     }
186   }
187 }
```