µracoli Manual

Version foo

Generated by Doxygen 1.8.10

Sat Aug 29 2015 08:07:16

Contents

1 Main Page

µracoli stands for ***microcontroller radio communications library*** and is a package that demonstrates the capabilities and usage of Atmels IEEE-802.15.4 radio transceivers. Currently supported are

- AT86RF230a, AT86RF230b
- AT86RF231
- AT86RF232
- AT86RF233
- AT86RF212
- ATmega128RFA1
- ATmega256RFR2, ATmega2564RFR2

in combination with Atmel AVR microcontrollers of the families ATmega, ATxmega and ATtiny.

The picture shows the software components of the µracoli project.

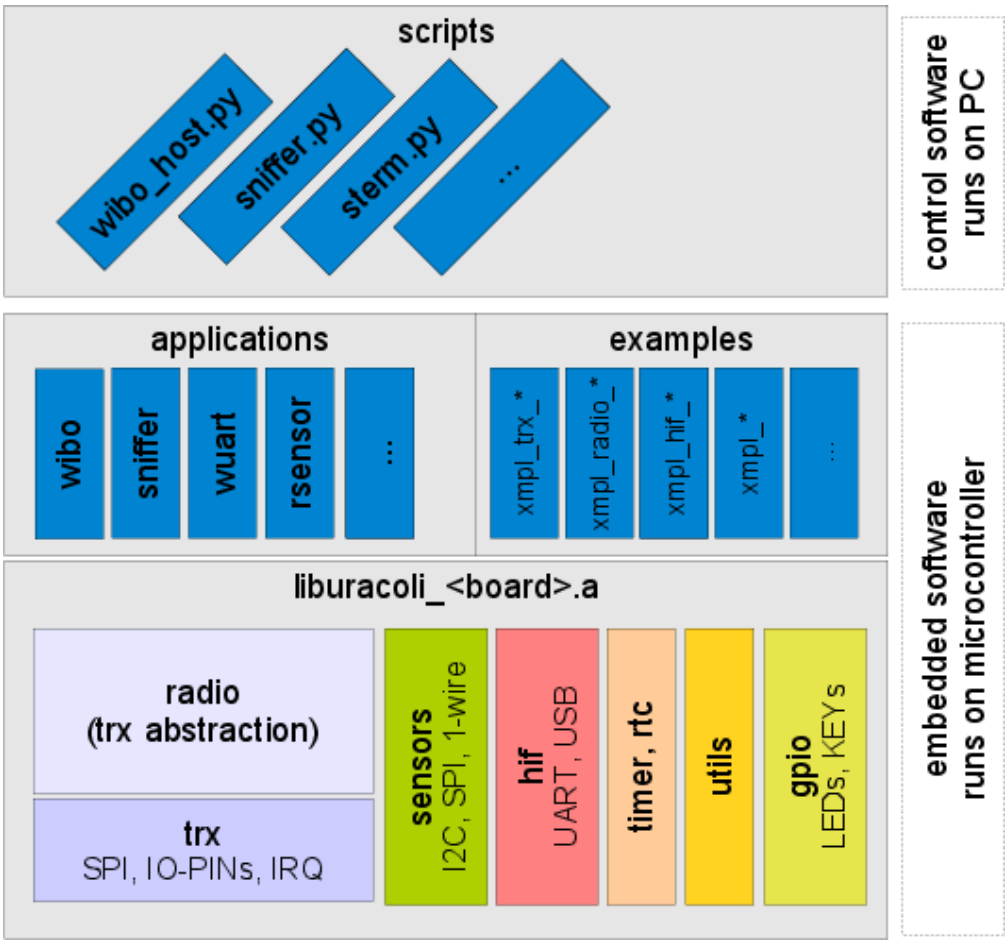


Figure 1: Software modules

The core component of µracoli is the **library** `liburacoli_<board>.a`. It contains the driver functions for the hardware components of the over 100 supported boards, e.g.

- low and high level drivers for the supported radio transceivers,
- drivers for several sensors with SPI-, one-wire- and I2C-interface,
- drivers for host interface communication,
- timer and rtc implementation and
- GPIO handling (e.g. LEDs and KEYs).

Using a library means, that only code that is called from the application is linked into the firmware image. This results in small programm sizes, leaving as much memory as possible for the application.

On top of the library there exist **examples** and **applications**, which are microcontroller applications that make use of the library-functions.While the `examples` are very simple programs, that serve just as illustraion of the use of the library functions, the `applications` implement more complex programs, that can be of general use as utility or tool.

In some cases there are specialised PC-applications, that communicate via a serial port with the microcontroller board. These programs are refered as **scripts**. since they are primarily written in the Python programming language.

µracoli supports over 100 **different PCBs** with various combinations of microcontrollers, transceivers and sensors. Therefore a generic platform abstraction concept was developped, which allows the easy definition and integration of new PCBs into the project.

2 User Guide

This Manual conains the following sections:

- Getting Started
  How to do the first steps with µracoli ?
- Boards and Modules
  A list of supported hardware.
- Library
  Library concept
- Examples
  A set of rather basic examples.
- Applications
  Out of the box applications like wireless uart, sniffer, ...
- HOWTOs
  Detailed information about the development process.
- Epilogue
  Release notes, License, Acknowledgements.

2.1 Getting Started

2.1.1 Overview

- Quick Start
  Jumpstart using µracoli with your hardware.
- Architecture
  Some words about the concepts of µracoli.
- Package Contents
  Contents of the source package.
- The Source Code Package (todo: split to existing pages)
  Overview of the package contents.
- Prerequisites
  What is needed to start with µracoli?
- Compiling the Library
  How the Library is built for specific boards.
- Using Examples and Applications
  Compiling and using the provided applications.
- Creating Own Projects
  Creating own applicationw with liburacoli

2.1.2 Quick Start

- Install a toolchain (see Prerequisites)
- Select your board from Boards and Modules, rember the target name.
- have avr-gcc and make in your PATH.
- execute "`make -C src/wuart <target>`"

- Flash @ "src/bin/wuart_radiofaro.hex"
- connect a terminal to your board
- press reset and see in the terminal the boot message
- Flash the second board and connect it to a terminal too
- the letters typed in one window should be displayed in the second terminal window the second terminal window
- todo add pictures
- todo add links to Howto sections.

2.1.3 Architecture

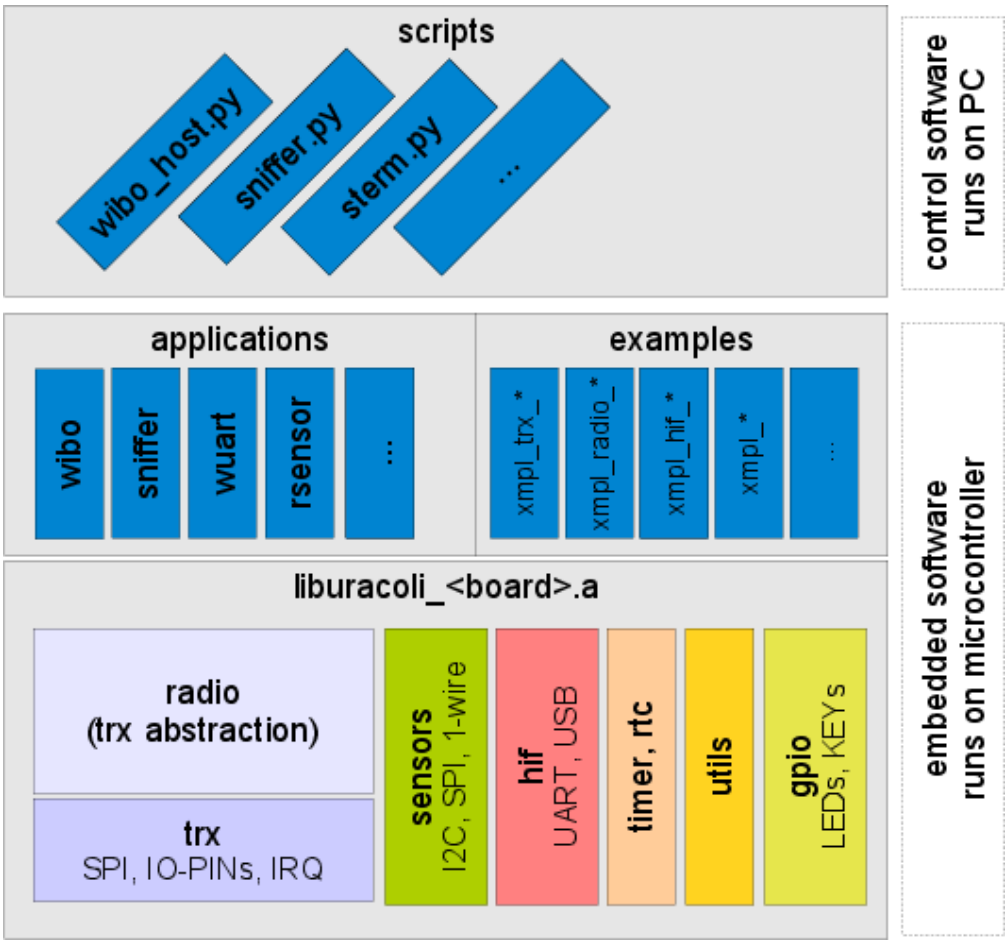


Figure 2: Software modules

2.1.4 Package Contents

```
uracoli-<version>
|-- doc
|   `-- search
|-- scripts
`-- src
    |-- as6
    |   `-- halimbawa
    |-- examples
    |-- inc
    |   |-- boards
```

```
|   `-- sensors
|-- rsensor
|-- sniffer
|-- uracoli
|-- wibo
|   |-- wibo
|   |-- wibohost
|   `-- wibotest
`-- wuart
```

2.1.5 Prerequisites

- Two or more of the supported transceiver boards (see Boards and Modules) are needed.
- In order to flash the compiled firmware into the micro controller, an hardware programmer like AVRISP, JTAGICE mkII, AVR-Dragon or a similiar tool is needed.
- A working AVR toolchain is required. The GNU AVR tools (or WinAVR on Windows) are used for compiling the firmware. The programs avrdude, avarice, avr-gdb (or AVR-Studio on Windows) are used to flash and debug the compiled applications.
- Additionally you probably want to install the python programming language with a version before .0 with the module pyserial in order to easily run and create PC applications that interact over the serial port with a radio module.
- If you want to build all libraries and applications from scratch, scons is needed. Building the documentation requires doxygen and graphviz.

Todo: clean / remove the link, add Toolchain description in HowTo. Some more detailed information about installing the software can be found at http://uracoli.nongnu.org/gettingstarted.html.

2.1.6 Compiling the Library

```
cd uracoli-<version>/src/uracoli
make [help|list|all|<board>]
```

- "make help" : display the usage message
- "make list" : display all supported boards.
- "make all" : compile all libraries lib/liburacoli_*.a
- "make <board>" : compile the library lib/liburacoli_<board>.a

2.1.7 Using Examples and Applications

The examples are minimal C-Programms, that illustrate the usage of the liburacoli functions. For more details refer to section Compiling the Examples.

The applications implement more complex functions and can be thought as tools, like a wireless UART or a sniffer. For more details refer to section Applications. For some of the applications a supplementary PC-software is required, e.g. sniffer.py for the firmware sniffer_<board>.hex. In order to be platform independent, the PC-software is written in Python.

2.1.8 Creating Own Projects

The library is used by the

- examples and
- applications or it can be used in

- own applications.

The Library Reference documents the functions seperately.

As a starting point for an own application, one of the examples can be used.

Make a directory, e.g. myproject in "uracoli-<version>/src"

Start with writing an initial C-File and store it as myproject.c:

```
#include <stdio.h>
#include "board.h"
#include "transceiver.h"
#include "radio.h"

uint8_t RxFrame[MAX_FRAME_SIZE];

int main(void)
{
  radio_init(RxFrame, sizeof(RxFrame));
  while(1)
  {
  }
  return 0;
}
```

Now the C-File can be compiled and linked against liburacoli. In order to get the compiler and linker flags, execute the compiling of one of the examples and copy the command line from there:

```
make -C ../examples/  -n -f xmpl_hif.mk rdk230
...
avr-gcc -Wall -Wundef -Os -g -ffunction-sections -fdata-sections -std=c99 -mmcu=atmega1281 -Wa,-adhlns=../
      build/xmpl_hif_rdk230.lst -Drdk230 -DF_CPU=8000000UL -DAPP_NAME="\"xmpl_hif\"" -I../inc -I. -c -o ../build/
      xmpl_hif_rdk230.o xmpl_hif.c
avr-gcc -o ../bin/xmpl_hif_rdk230.elf -Wall -Wundef -Os -g -ffunction-sections -fdata-sections -std=c99 -
      mmcu=atmega1281 -Wa,-adhlns=../build/xmpl_hif_rdk230.o -Drdk230 -DF_CPU=8000000UL -DAPP_NAME="\"xmpl_hif\"" -I
      ../inc -I. ../build/xmpl_hif_rdk230.o -L../lib -Wl,--gc-sections -luracoli_rdk230
avr-objcopy -O ihex ../bin/xmpl_hif_rdk230.elf ../bin/xmpl_hif_rdk230.hex
...
```

The own project can then be build with the command line

```
avr-gcc -Os -g \
        -std=c99 -mmcu=atmega1281 \
        -Drdk230 -DF_CPU=8000000UL \
        -I../inc -I. \
        -o myproject.elf \
        myproject.c \
        -L../lib -Wl,--gc-sections \
        -luracoli_rdk230
```

The file myproject.elf contains the firmware and can be flashed on the tartet board.

2.1.9 The Source Code Package (todo: split to existing pages)

Overview

This package contains the uracoli-source code. It consists of the library, various example programms that illustrate the usage of the library functions and some end user applications.

Directory	Content
src/	source code of the μracoli library .
inc/	The library header file.
wuart/	The wireless UART application xxx .
sniffer/	The source code of the sniffer firmware.

wibo/	Wireless bootloader source code, host application and examples.
xmpl/	Some example applications, that illustrate how to use the µracoli functions.
as6/	Atmel Studio 6 project template.

Prerequisites

In order to use the libraries and applications you need GNU-make and the GNU-AVR toolchain, consisting of compiler, linker, avr-libc and a hardware programming tool.

AVR toolchain

Under *Windows* install Atmel Studio 6. For older Atmel MCUs (up to Atmega128RFA1) the last version of WinAVR is fine too.

On *Linux* install the following packages with the packet manager of the distribution. In Ubuntu or Debian this can be achieved with the following command line:

```
sudo apt-get install avr-libc binutils-avr gcc-avr avrdude avarice gdb-avr
```

- avr-gcc - The AVR GCC C-compiler
- avr-binutils - Linker, Object File Converter, ...
- avr-libc - Standard C library which provides a good set of C standard functions
- avrdude - Tool for programming to the internal Flash memory and/or EEPROM.
- avr-gdb - GNU debugger to debug AVR programs
- AVaRICE - Tool to interfaces avr-gdb with the Atmel JTAG ICE hardware debugger.

A detailed installation description is available here.

Python

Python version 2.7.x is required to run the μracoli tools like sterm or the sniffer interface.

Under *Windows* intstall the MSI package for Python 2.7.9 from http://www.python.org. This package has pip already included, so the installation of further packages is rather simple.

Now run `pip install serial` with administrator privilres.

Under *Linux* install Python and Python-pip with the package manager of your distribution.

Additionally run `pip install serial` as super user.

The libraries can be build with make. In order to get an overview, if your board is supported, type

```
make -C src/ list
```

The libraries for e.g. the "radiofaro" board, are build with the command:

```
make -C src radiofaro
```

This will create the directory +lib+ and the uracoli-library for radiofaro.

```
$ls lib/
liburacoli_radiofaro.a
```

Scritable Terminal Program

This python script +wuart/sterm.py+ is a usefull addon for debugging wireless applications. How to use the script is described in section sterm

Sniffer Firmware

With the sniffer firmware the frames exchanged in an IEEE 802.15.4 network can be captured and transmitted over the serial interface to the PC. This firmware is intended for use with a the Python script +sniffer.py+ and http:www.wireshark.org[wireshark]. The script together with the documentation, which explains the sniffer setup, can be found in the package uracoli-sniffer-<version>.zip on http://uracoli.nongnu.org/download.html[].

The sniffer firmware can be compiled with the commands

```
make -C src mysnifferboard
make -C sniffer mysnifferboard
```

For which boards the sniffer application is available, you can find out with the command: make -C sniffer list

Wireless Bootloader

The wireless bootloader (WiBo) is an application that resides in the bootloader section of the AVR and therefore it can erase and rewrite the programm flash memory. In Difference to a regular bootloader, WiBo receives its data over the RF link. WiBo modifies the flash directly, therefore no special backup flash memory, like in other over the air upgrade (OTA) solutions, is required on the transceiver board.

A detailed description how WiBo is used can be found in section wibo.

Examples

The example source code can be found in the directory +xmpl/+. This simple example programms are thought as starters for your application.

- `xmpl_leds.c` Example use of the LED macros
- `xmpl_key_events.c` Example for key event processing with a single key
- `xmpl_keys.c` Example use of the KEY macros
- `xmpl_hif.c` Example for use of the HIF functions
- `xmpl_hif_echo.c` Example that implements HIF echo, usefull to test the HIF troughput
- `xmpl_timer.c` Example for using the timer macros
- `xmpl_dbg.c` Example for use of the DBG_XXX macros
- `xmpl_linbuf_tx.c` Example use of the buffer functions
- `xmpl_trx_base.c` Example for accessing the transceiver
- `xmpl_trx_echo.c` Example for echoing received frames
- `xmpl_trx_rxaack.c` Example for receiving frames in rx_aack mode
- `xmpl_trx_rx.c` Example for receiving frames
- `xmpl_trx_txaret.c` Example for transmitting frames in tx_aret mode
- `xmpl_trx_tx.c` Example for transmitting frames
- `xmpl_radio_range.c` Example use of the radio and ioutil functions for a simple range test
- `xmpl_radio_stream.c` Example use of the radio stream functions

The example firmware can be build with the following commands:

```
make -C src myboard
make -C xmpl -f xmpl_<myexample>.mk myboard
```

Note: Some of the examples are not available on all boards, due to the lack of some hardware features, e.g. if the LEDs, the KEYs or/and the HIF is absent. A list of supported boards of the example can be obtained by:

make -C xmpl -f xmpl_<myexample>.mk list

2.2 Boards and Modules

2.2.1 any2400

Board #1: A.N. Solutions ANY Brick

Parameters

- Build Target: `any2400`
- Include file: `board_any.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, led, lm73, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_lm73, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.2 any2400st

Board #2: A.N. Solutions ANY Stick

Parameters

- Build Target: `any2400st`
- Include file: `board_any.h`
- Baudrate: `38400`
- MMCU: `atmega1281`

- F_CPU: `8000000UL`
- Provides: `hif, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩
stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.3 any900

Board #3: A.N. Solutions ANY Brick

Parameters

- Build Target: `any900`
- Build Aliases: any2400
- Include file: `board_any.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, led, lm73, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_lm73, xmpl_radio↩
_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx↩
_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.4 any900st

Board #4: A.N. Solutions ANY Stick

Parameters

- Build Target: `any900st`
- Build Aliases: any2400st
- Include file: `board_any.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.5 atrcb256rfr2xpro

Board #5: Atmel ATmega256RFR2 ZigBit Xplained Pro Extension

Parameters

- Build Target: `atrcb256rfr2xpro`
- Include file: `board_derfa.h`
- Baudrate: `57600`
- MMCU: `atmega256rfr2`
- F_CPU: `16000000UL`
- Provides: `hif, key, led, mcu_t, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.6 atzb256rfr2xpro

Board #6: Atmel ATmega256RFR2 ZigBit Xplained Pro Extension

Parameters

- Build Target: `atzb256rfr2xpro`
- Build Aliases: atrcb256rfr2xpro
- Include file: `board_derfa.h`
- Baudrate: `57600`
- MMCU: `atmega256rfr2`
- F_CPU: `16000000UL`
- Provides: `hif, key, led, mcu_t, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.7 atzbx212bxpro

Board #7: Atmel ZigBit (Module, USB Stick, Xplained Pro Extension)

Parameters

- Build Target: `atzbx212bxpro`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.8 atzbx233usb

Board #8: Atmel ZigBit (Module, USB Stick, Xplained Pro Extension)

Parameters

- Build Target: `atzbx233usb`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3u`
- F_CPU: `32000000UL`
- Provides: `led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

rsensor, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_rtc, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.9 atzbx233xpro

Board #9: Atmel ZigBit (Module, USB Stick, Xplained Pro Extension)

Parameters

- Build Target: `atzbx233xpro`
- Build Aliases: atzbx212bxpro
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`

- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.10 bat

Board #10: AirDMX remote node Bat, battery powered

Parameters

- Build Target: `bat`
- Include file: `board_airdmx.h`
- Baudrate: `57600`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xf7:m -U hfuse:w:0x9a:m -U efuse:w:0xfe:m`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.11 bitbean

Board #11: Colorado Micro Devices, BitBean (ZigBit ATZB-24-A2)

Parameters

- Build Target: `bitbean`
- Include file: `board_wdba1281.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Fuses: `-U lfuse:w:0xe2:m -U hfuse:w:0x99:m -U efuse:w:0xff:m`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.12 cbb212

Board #12: REB Controller Base Board with REB23x/REB212 attached

Parameters

- Build Target: `cbb212`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, key, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.13 cbb230

Board #13: REB Controller Base Board with REB23x/REB212 attached

Parameters

- Build Target: `cbb230`
- Build Aliases: cbb230b, cbb231, cbb212, cbb232, cbb233
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, key, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.14 cbb230b

Board #14: REB Controller Base Board with REB23x/REB212 attached

Parameters

- Build Target: `cbb230b`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`

- F_CPU: `8000000UL`
- Provides: `hif, i2c, key, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.15 cbb231

Board #15: REB Controller Base Board with REB23x/REB212 attached

Parameters

- Build Target: `cbb231`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, key, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.16 cbb232

Board #16: REB Controller Base Board with REB23x/REB212 attached

Parameters

- Build Target: `cbb232`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, key, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.17 cbb233

Board #17: REB Controller Base Board with REB23x/REB212 attached

Parameters

- Build Target: `cbb233`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega256a3`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, key, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.18 derfa1

Board #18: Dresden Elektronik Radio Module deRFmega128-22A001

Parameters

- Build Target: `derfa1`
- Include file: `board_derfa.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_t, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.19 derfn128

Board #19: Dresden Elektronik Radio Module deRFmega128-22A/M{00} on deRFnode, USB

Parameters

- Build Target: `derfn128`
- Include file: `board_derfa.h`
- Baudrate: `57600`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xc6:m -U hfuse:w:0x19:m -U efuse:w:0xfe:m`
- Provides: `hif, i2c, isl29020, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_isl29020, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.20 derfn128u0

Board #20: Dresden Elektronik Radio Module deRFmega128-22A/M{00} on deRFnode, USB

Parameters

- Build Target: `derfn128u0`
- Build Aliases: derfn128
- Include file: `board_derfa.h`
- Baudrate: `57600`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xc6:m -U hfuse:w:0x19:m -U efuse:w:0xfe:m`
- Provides: `hif, i2c, isl29020, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_isl29020, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.21 derfn256u0

Board #21: Dresden Elektronik Radio Module deRFmega256-23M{00,10,12} on deRFnode, UART0 (X5)

Parameters

- Build Target: `derfn256u0`
- Build Aliases: derfn256u0pa
- Include file: `board_derfa.h`

- Baudrate: `57600`
- MMCU: `atmega256rfr2`
- F_CPU: `16000000UL`
- Provides: `hif, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_↩ rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.22 derfn256u0pa

Board #22: Dresden Elektronik Radio Module deRFmega256-23M{00,10,12} on deRFnode, UART0 (X5)

Parameters

- Build Target: `derfn256u0pa`
- Include file: `board_derfa.h`
- Baudrate: `57600`
- MMCU: `atmega256rfr2`
- F_CPU: `16000000UL`
- Provides: `hif, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_↩ rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.23 derftorcbrfa1

Board #23: Dresden Elektronik deRFtoRCB Adapter for ATmega128RFA1

Parameters

- Build Target: `derftorcbrfa1`
- Include file: `board_rbbrfa1.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.24 dracula

Board #24: AirDMX gateway Dracula

Parameters

- Build Target: `dracula`
- Include file: `board_airdmx.h`
- Baudrate: `57600`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xf7:m -U hfuse:w:0x9a:m -U efuse:w:0xfe:m`
- Provides: `hif, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_↩ rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.25 ibdt212

Board #25: IBDT212 Hardware

Parameters

- Build Target: `ibdt212`
- Include file: `board_ibdt.h`
- Baudrate: `38400`
- MMCU: `atmega644`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.26 ibdt231

Board #26: IBDT231 Hardware

Parameters

- Build Target: `ibdt231`
- Include file: `board_ibdt.h`
- Baudrate: `38400`
- MMCU: `atmega644`

- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.27 ibdt232

Board #27: IBDT232 Hardware

Parameters

- Build Target: `ibdt232`
- Include file: `board_ibdt.h`
- Baudrate: `38400`
- MMCU: `atmega644p`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.28 icm230_11

Board #28: In-Cirquit radio module, version 1.1, 1.2, AT86RF230a/b

Parameters

- Build Target: `icm230_11`
- Build Aliases: icm230_12a, icm230_12b
- Include file: `board_ict_11.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.29 icm230_12a

Board #29: In-Cirquit radio module, version 1.1, 1.2, AT86RF230a/b

Parameters

- Build Target: `icm230_12a`
- Include file: `board_ict_11.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.30 icm230_12b

Board #30: In-Cirquit radio module, version 1.1, 1.2, AT86RF230a/b

Parameters

- Build Target: `icm230_12b`
- Include file: `board_ict_11.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.31 icm230_12c

Board #31: In-Cirquit radio stick/module, version 1.2a (RF230 RevB) [tarnished finish & AtMega128]

Parameters

- Build Target: `icm230_12c`
- Include file: `board_ict_11.h`
- Baudrate: `38400`
- MMCU: `atmega128`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.32 ics230_11

Board #32: In-Cirquit radio stick, version 1.1

Parameters

- Build Target: `ics230_11`
- Include file: `board_ict_11.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.33 ics230_12

Board #33: In-Cirquit radio stick/module, version 1.2a (RF230 RevB) [tarnished finish & AtMega128]

Parameters

- Build Target: `ics230_12`
- Build Aliases: icm230_12c
- Include file: `board_ict_11.h`
- Baudrate: `38400`
- MMCU: `atmega128`

- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.34 ict230

Board #34: In-Cirquit radio stick/module, version 1.0

Parameters

- Build Target: `ict230`
- Include file: `board_ict230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.35 im240a

Board #35: IMST GmbH, WiMOD im240a Module

Parameters

- Build Target: `im240a`
- Include file: `board_wimod.h`
- Baudrate: `None`
- MMCU: `atmega328`
- F_CPU: `8000000UL`
- Fuses: `-U lfuse:w:0xe2:m -U hfuse:w:0xd4:m -U efuse:w:0x06:m`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.36 im240a_eval

Board #36: IMST GmbH, WiMOD im240a Development Board

Parameters

- Build Target: `im240a_eval`
- Include file: `board_wimod.h`
- Baudrate: `None`
- MMCU: `atmega328`
- F_CPU: `8000000UL`
- Fuses: `-U lfuse:w:0xe2:m -U hfuse:w:0xd4:m -U efuse:w:0xfe:m`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.37 l3y

Board #37: 2564rfr2 low cost module

Parameters

- Build Target: `l3y`
- Include file: `board_derfa.h`
- Baudrate: `9600`
- MMCU: `atmega2564rfr2`
- F_CPU: `8000000UL`
- Fuses: `-U lfuse:w:0xe2:m -U hfuse:w:0x91:m -U efuse:w:0xfe:m`
- Provides: `hif, led, mcu_t, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.38 lgee231

Board #38: DIY Board by Daniel Thiele, w/ accelerometer, breakout board and UART.

Parameters

- Build Target: `lgee231`
- Include file: `board_lgee.h`
- Baudrate: `None`
- MMCU: `atmega88`
- F_CPU: `8000000UL`
- Provides: `led, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.39 lgee231_v2

Board #39: DIY board by Daniel Thiele, w/ accelerometer, w/o breakout board.

Parameters

- Build Target: `lgee231_v2`
- Include file: `board_lgee.h`
- Baudrate: `None`
- MMCU: `atmega88`
- F_CPU: `8000000UL`
- Provides: `led, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.40 midgee

Board #40: IBDT Midgee

Parameters

- Build Target: `midgee`
- Include file: `board_midgee.h`
- Baudrate: `None`
- MMCU: `atmega88p`
- F_CPU: `8000000UL`

- Provides: `led, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.41 mnb900

Board #41: Meshnetics MeshBean WDB-A1281 and MNZB-900 development boards

Parameters

- Build Target: `mnb900`
- Include file: `board_wdba1281.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `ds18b20, hif, i2c, led, lm73, mcu_vtg, ow, rtc, sensors, tmr, trx, trxvtg, tsl2550`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_lm73, xmpl_ow, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret, xmpl_tsl2550

2.2.42 muse231

Board #42: IBDT Multisensor Board

Parameters

- Build Target: `muse231`
- Include file: `board_muse231.h`
- Baudrate: `None`
- MMCU: `atmega88pa`
- F_CPU: `8000000UL`
- Provides: `led, mcu_vtg, sensors, sht21_rh, sht21_t, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.43 musell232

Board #43: IBDT Musell ATmega88PA+AT86RF232

Parameters

- Build Target: `museII232`
- Include file: `board_museII232.h`
- Baudrate: `None`
- MMCU: `atmega328p`
- F_CPU: `8000000UL`
- Provides: `led, rtc, tmr, trx`

Applications

rsensor, wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_rtc, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.44 musellrfa

Board #44: IBDT Musell ATmega128RFA1

Parameters

- Build Target: `museIIrfa`
- Include file: `board_museIIrfa.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.45 pinoccio

Board #45: Pinoccio - the ecosystem for the internet of things

Parameters

- Build Target: `pinoccio`
- Include file: `board_derfa.h`
- Baudrate: `115200`
- MMCU: `atmega256rfr2`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xf7:m -U hfuse:w:0xda:m -U efuse:w:0xf5:m`
- Provides: `hif, led, mcu_t, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.46 psk212

Board #46: Atmel Packet Sniffer Kit, STK541 with RCB for AT86RF{212,23x}

Parameters

- Build Target: `psk212`
- Include file: `board_stk541.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.47 psk230

Board #47: Atmel Packet Sniffer Kit, STK541 with RCB for AT86RF{212,23x}

Parameters

- Build Target: `psk230`
- Build Aliases: psk230b, psk231, psk212, psk232, psk233
- Include file: `board_stk541.h`
- Baudrate: `None`
- MMCU: `atmega1281`

- F_CPU: `8000000UL`
- Provides: `hif, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.48 psk230b

Board #48: Atmel Packet Sniffer Kit, STK541 with RCB for AT86RF{212,23x}

Parameters

- Build Target: `psk230b`
- Include file: `board_stk541.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.49 psk231

Board #49: Atmel Packet Sniffer Kit, STK541 with RCB for AT86RF{212,23x}

Parameters

- Build Target: `psk231`
- Include file: `board_stk541.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.50 psk232

Board #50: Atmel Packet Sniffer Kit, STK541 with RCB for AT86RF{212,23x}

Parameters

- Build Target: `psk232`
- Include file: `board_stk541.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.51 psk233

Board #51: Atmel Packet Sniffer Kit, STK541 with RCB for AT86RF{212,23x}

Parameters

- Build Target: `psk233`
- Include file: `board_stk541.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_vtg, rtc, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.52 radiofaro

Board #52: RadioFaro, Arduino like board with deRFmega128-22A001

Parameters

- Build Target: `radiofaro`
- Include file: `board_derfa.h`
- Baudrate: `57600`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xf7:m -U hfuse:w:0x9a:m -U efuse:w:0xfe:m`
- Provides: `hif, led, mcu_t, mcu_vtg, rtc, sensors, tmr, trx, trxvtg`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.53 radiofaro_v1

Board #53: RadioFaro, Arduino like board with deRFmega128-22A001

Parameters

- Build Target: `radiofaro_v1`
- Include file: `board_derfa.h`
- Baudrate: `38400`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.54 raspbee

Board #54: Dresden Elektronik Raspberry Pi Module

Parameters

- Build Target: `raspbee`
- Include file: `board_derfa.h`
- Baudrate: `38400`
- MMCU: `atmega256rfr2`
- F_CPU: `8000000UL`

- Provides: `hif, led, rtc, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_rtc, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.55 ravrf230a

Board #55: Atmel Raven Board w/ AT86RF230A/B

Parameters

- Build Target: `ravrf230a`
- Build Aliases: ravrf230b
- Include file: `board_ravrf.h`
- Baudrate: `9600`
- MMCU: `atmega1284p`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.56 ravrf230b

Board #56: Atmel Raven Board w/ AT86RF230A/B

Parameters

- Build Target: `ravrf230b`
- Include file: `board_ravrf.h`
- Baudrate: `9600`
- MMCU: `atmega1284p`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.57 rbb128rfa1

Board #57: Dresden Elektronik Breakout Board, with RCB for ATmega128RFA1

Parameters

- Build Target: `rbb128rfa1`
- Include file: `board_rbbrfa1.h`
- Baudrate: `38400`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `hif, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.58 rbb212

Board #58: Dresden Elektronik Breakout Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `rbb212`
- Include file: `board_rbb.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.59 rbb230

Board #59: Dresden Elektronik Breakout Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `rbb230`
- Build Aliases: rbb230b, rbb231, rbb212, rbb232, rbb233
- Include file: `board_rbb.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.60 rbb230b

Board #60: Dresden Elektronik Breakout Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `rbb230b`
- Include file: `board_rbb.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.61 rbb231

Board #61: Dresden Elektronik Breakout Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `rbb231`
- Include file: `board_rbb.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.62 rbb232

Board #62: Dresden Elektronik Breakout Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `rbb232`
- Include file: `board_rbb.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.63 rbb233

Board #63: Dresden Elektronik Breakout Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `rbb233`
- Include file: `board_rbb.h`
- Baudrate: `38400`

- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.64 rdk212

Board #64: Radio Controller Board by Atmel and Dresden Elektronik

with AT86RF{212,23x}

Parameters

- Build Target: `rdk212`
- Include file: `board_rdk230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.65 rdk230

Board #65: Radio Controller Board by Atmel and Dresden Elektronik

with AT86RF{212,23x}

Parameters

- Build Target: `rdk230`
- Build Aliases: rdk230b, rdk231, rdk212, rdk232, rdk233
- Include file: `board_rdk230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.66 rdk230b

Board #66: Radio Controller Board by Atmel and Dresden Elektronik

with AT86RF{212,23x}

Parameters

- Build Target: `rdk230b`
- Include file: `board_rdk230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.67 rdk231

Board #67: Radio Controller Board by Atmel and Dresden Elektronik

with AT86RF{212,23x}

Parameters

- Build Target: `rdk231`
- Include file: `board_rdk230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.68 rdk232

Board #68: Radio Controller Board by Atmel and Dresden Elektronik

with AT86RF{212,23x}

Parameters

- Build Target: `rdk232`
- Include file: `board_rdk230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.69 rdk233

Board #69: Radio Controller Board by Atmel and Dresden Elektronik

with AT86RF{212,23x}

Parameters

- Build Target: `rdk233`
- Include file: `board_rdk230.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.70 rose231

Board #70: IBDT Rocket Sensor Board

Parameters

- Build Target: `rose231`
- Include file: `board_rose231.h`
- Baudrate: `None`
- MMCU: `atmega328p`
- F_CPU: `8000000UL`

- Provides: `hmc5883l, led, sensors, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.71 rzusb

Board #71: Atmel Raven USB Stick with AT86RF230 Rev. B

Parameters

- Build Target: `rzusb`
- Include file: `board_rzusb.h`
- Baudrate: `None`
- MMCU: `at90usb1287`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.72 sparcrfa1

Board #72: Sparcfun RFA1 Dev. Board by J. Lindblom

Parameters

- Build Target: `sparcrfa1`

- Include file: `board_derfa.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xff:m -U hfuse:w:0xda:m -U efuse:w:0xf5:m`
- Provides: `hif, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.73 stb128rfa1

Board #73: Dresden Elektronik Sensor Terminal Board with RCB128RFA1

Parameters

- Build Target: `stb128rfa1`
- Include file: `board_stbrfa1.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `hif, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_↩rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.74 stb212

Board #74: Dresden Elektronik Sensor Terminal Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `stb212`
- Include file: `board_stb.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.75 stb230

Board #75: Dresden Elektronik Sensor Terminal Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `stb230`
- Build Aliases: stb230b, stb231, stb212, stb232, stb233
- Include file: `board_stb.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.76 stb230b

Board #76: Dresden Elektronik Sensor Terminal Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `stb230b`
- Include file: `board_stb.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.77 stb231

Board #77: Dresden Elektronik Sensor Terminal Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `stb231`
- Include file: `board_stb.h`
- Baudrate: `None`
- MMCU: `atmega1281`

- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.78 stb232

Board #78: Dresden Elektronik Sensor Terminal Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `stb232`
- Include file: `board_stb.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.79 stb233

Board #79: Dresden Elektronik Sensor Terminal Board with RCB for AT86RF{212,23x}

Parameters

- Build Target: `stb233`
- Include file: `board_stb.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, key, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_key_events, xmpl_keys, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.80 stb256rfr2

Board #80: Sensor Terminal Board with Atmel RCB256RFR2 Radio Controller Board

Parameters

- Build Target: `stb256rfr2`
- Include file: `board_stbrfa1.h`
- Baudrate: `None`
- MMCU: `atmega256rfr2`
- F_CPU: `8000000UL`
- Provides: `hif, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.81 stkm16

Board #81: STK500 with ATmega16 and AT86RF230 radio extender board

Parameters

- Build Target: `stkm16`
- Include file: `board_stkm16.h`
- Baudrate: `115200`
- MMCU: `atmega16`
- F_CPU: `3686400UL`
- Fuses: `-U lfuse:w:0xef:m -U hfuse:w:0x91:m`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.82 stkm8

Board #82: STK500 with ATmega8 and AT86RF230 radio extender board

Parameters

- Build Target: `stkm8`
- Include file: `board_stkm8.h`
- Baudrate: `None`
- MMCU: `atmega8`
- F_CPU: `8000000UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, selftest, testlib, wgpio, wibo, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.83 tiny230

Board #83: DIY Board by Joerg Wunsch with ATtiny(44,84) and AT86RF(230,231)

Parameters

- Build Target: `tiny230`
- Build Aliases: tiny231
- Include file: `board_tiny230.h`
- Baudrate: `None`
- MMCU: `attiny84`
- F_CPU: `8000000UL`
- Provides: `led, ledps, sensors, tmr, trx`

Applications

wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.84 tiny231

Board #84: DIY Board by Joerg Wunsch with ATtiny(44,84) and AT86RF(230,231)

Parameters

- Build Target: `tiny231`
- Include file: `board_tiny230.h`
- Baudrate: `None`

- MMCU: `attiny84`
- F_CPU: `8000000UL`
- Provides: `led, ledps, sensors, tmr, trx`

Applications

wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.85 wdba1281

Board #85: Meshnetics MeshBean WDB-A1281 and MNZB-900 development boards

Parameters

- Build Target: `wdba1281`
- Build Aliases: mnb900
- Include file: `board_wdba1281.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `ds18b20, hif, i2c, led, lm73, mcu_vtg, ow, rtc, sensors, tmr, trx, trxvtg, tsl2550`

Applications

diag, rdiag, rsensor, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_lm73, xmpl_ow, xmpl_radio_range, xmpl_radio_stream, xmpl_rtc, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret, xmpl_tsl2550

2.2.86 wprog

Board #86: WProg

Parameters

- Build Target: `wprog`
- Include file: `board_derfa.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `8000000UL`
- Provides: `hif, led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_↩ rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.87 xma1u233xpro

Board #87: Atmel ZigBit ATZB-RF-233-1-C XPRO Extension connected to XMEGA A1U Xplained Pro

Parameters

- Build Target: `xma1u233xpro`
- Include file: `board_cbb2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega128a1u`
- F_CPU: `8000000UL`
- Provides: `keys, led, mcu_vtg, rtc, sensors, tmr, trx`

Applications

rsensor, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_rtc, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.88 xme5rz212

Board #88: Atmel RZ600 Stick plugged on J1 of Xmega-E5 Xplained

Parameters

- Build Target: `xme5rz212`
- Include file: `board_xme5rz2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega32e5`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, keys, led, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.89 xme5rz230

Board #89: Atmel RZ600 Stick plugged on J1 of Xmega-E5 Xplained

Parameters

- Build Target: `xme5rz230`
- Build Aliases: xme5rz231, xme5rz212
- Include file: `board_xme5rz2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega32e5`
- F_CPU: `8000000UL`

- Provides: `hif, i2c, keys, led, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.90 xme5rz231

Board #90: Atmel RZ600 Stick plugged on J1 of Xmega-E5 Xplained

Parameters

- Build Target: `xme5rz231`
- Include file: `board_xme5rz2xx.h`
- Baudrate: `38400`
- MMCU: `atxmega32e5`
- F_CPU: `8000000UL`
- Provides: `hif, i2c, keys, led, mcu_vtg, sensors, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_i2c, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_sensor, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.91 xxo

Board #91: Tic-Tac-Toe Hardware for Chemnitzer Linuxtage 2012

Parameters

- Build Target: `xxo`
- Include file: `board_derfa.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `1000000UL`
- Fuses: `-U lfuse:w:0x62:m -U hfuse:w:0x18:m -U efuse:w:0xfe:m`
- Provides: `led, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

wibo, wibotest

Examples

xmpl_dbg, xmpl_leds, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.92 zgbh212

Board #92: ATZGB.com evaluation board

Parameters

- Build Target: `zgbh212`
- Include file: `board_zgbl.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `7372800UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.93 zgbh230

Board #93: ATZGB.com evaluation board

Parameters

- Build Target: `zgbh230`
- Build Aliases: zgbh231, zgbh212
- Include file: `board_zgbl.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `7372800UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_↩ stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_↩ trx_tx, xmpl_trx_txaret

2.2.94 zgbh231

Board #94: ATZGB.com evaluation board

Parameters

- Build Target: `zgbh231`
- Include file: `board_zgbl.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `7372800UL`
- Provides: `hif, led, tmr, trx`

Applications

diag, rdiag, selftest, sniffer, testlib, wgpio, wibo, wibohost, wibotest, wuart

Examples

xmpl_dbg, xmpl_hif, xmpl_hif_echo, xmpl_leds, xmpl_linbuf_rx, xmpl_linbuf_tx, xmpl_radio_range, xmpl_radio_stream, xmpl_timer, xmpl_timer_callback, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.95 zgbl212

Board #95: ATZGB.com radio modules

Parameters

- Build Target: `zgbl212`
- Include file: `board_zgbl.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `trx`

Applications

Examples

xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.96 zgbl230

Board #96: ATZGB.com radio modules

Parameters

- Build Target: `zgbl230`
- Build Aliases: zgbl231, zgbl212
- Include file: `board_zgbl.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`

- Provides: `trx`

Applications

Examples

xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.97 zgbl231

Board #97: ATZGB.com radio modules

Parameters

- Build Target: `zgbl231`
- Include file: `board_zgbl.h`
- Baudrate: `None`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `trx`

Applications

Examples

xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.98 zgbt1281a2nouart

Board #98: Meshnetics Zigbit A2, no UART

Parameters

- Build Target: `zgbt1281a2nouart`
- Include file: `board_zgbta2.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`

- Provides: `tmr, trx`

Applications

Examples

xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.99 zgbt1281a2uart0

Board #99: Meshnetics Zigbit A2, using UART0 (via ISP connector)

Parameters

- Build Target: `zgbt1281a2uart0`
- Include file: `board_zgbta2.h`
- Baudrate: `38400`
- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.100 zgbt1281a2uart1

Board #100: Meshnetics Zigbit A2, using UART1

Parameters

- Build Target: `zgbt1281a2uart1`
- Include file: `board_zgbta2.h`
- Baudrate: `38400`

- MMCU: `atmega1281`
- F_CPU: `8000000UL`
- Provides: `hif, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.2.101 zigduino

Board #101: Zigduino made by Logos Electromechanical LLC

Parameters

- Build Target: `zigduino`
- Include file: `board_derfa.h`
- Baudrate: `None`
- MMCU: `atmega128rfa1`
- F_CPU: `16000000UL`
- Fuses: `-U lfuse:w:0xff:m -U hfuse:w:0xda:m -U efuse:w:0xf5:m`
- Provides: `hif, mcu_t, mcu_vtg, sensors, tmr, trx`

Applications

selftest, sniffer, testlib, wibohost, wuart

Examples

xmpl_hif, xmpl_hif_echo, xmpl_sensor, xmpl_timer, xmpl_trx_base, xmpl_trx_echo, xmpl_trx_rx, xmpl_trx_rxaack, xmpl_trx_tx, xmpl_trx_txaret

2.3 Library

2.3.1 Overview

The µracoli-library consists of several modules and drivers for hardware components. The modules are isolated and can be used sperately, that means if your application just requires the radio functionality from µracoli, none of the other modules is linked.

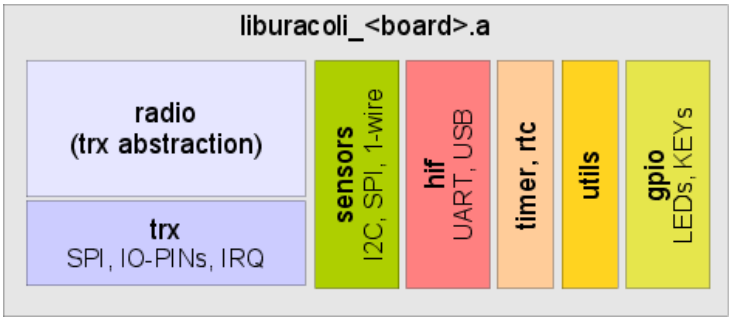


Figure 3: Library Overview

For each of the boards supported by µracoli, a seperate library is build. In the Board Definitions the currently supported boards can be found.

2.4 Examples

2.4.1 Overview

The µracoli examples illustrate the use of the various library functions.

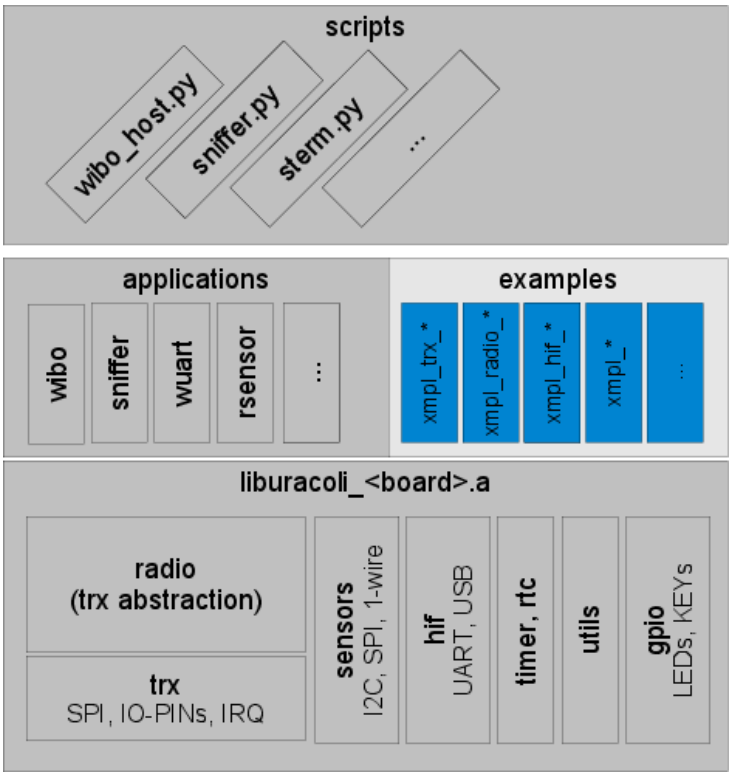


Figure 4: Examples Overview

This simple programms operate the LEDs, the HIF or the radio transceiver. They can be used either for testing the hardware or as starting point for own applications.

Compiling the Examples

Atmel Studio 6 Project Template

List of Examples

2.4.2 Compiling the Examples

The examples are located in the directory `uracoli-<version>/src/examples/`

Each example consists of a C source file, a GNUMakefile that is used for command line builds and an aps-file for Atmel AvrStudio . Here are the files, that belong to the example pgXmplHif.

```
xmpl.h ..................... common defines for all examples
xmpl_hif.c ................. example source file
xmpl_hif.aps ............... AVR Studio file
xmpl_hif.mk ................ GNU Makefile
```

Command Line Build (Linux, Windows)

The example can be build with command `make -f xmpl_hif.mk TARGET`. TARGET is either **all** (building this example for supported boards) or the name of the board to build for (see Boards and Modules).

Here is the log for building xmpl_hif.c for the board rdk230.

```
make -C install/xmpl/ -f xmpl_hif.mk rdk230
   make: Entering directory '[...]/install/xmpl'
   make -f xmpl_hif.mk BOARD=rdk230 MCU=atmega1281 F_CPU=8000000UL ./obj ./bin __xmpl_hif__
```

```
make[1]: Entering directory '[...]/install/xmpl'
avr-gcc -Wall -Wundef -Os -g -mmcu=atmega1281 -Wa,-adhlns=./obj/xmpl_hif_rdk230.lst -Drdk230 -D
avr-gcc -o bin/xmpl_hif_rdk230.elf -Wall -Wundef -Os -g -mmcu=atmega1281 -Wa,-adhlns=obj/xmpl_h
avr-objcopy -O ihex bin/xmpl_hif_rdk230.elf bin/xmpl_hif_rdk230.hex
make[1]: Leaving directory '[...]/install/xmpl'
make: Leaving directory '[...]/install/xmpl'
```

After make did run, the files `bin/xmpl_hif_rdk230.elf` and `bin/xmpl_hif_rdk230.hex` are created. The directory `./obj` stores temporary files that are created during the build.

AVR-Studio V4.x. Build

Double click on the xmpl_hif.aps and AvrStudio starts with the project file. In the next step select "Project/↩ Configuration Options", this opens the "Project Options" dialogue. In the section "General" there is a drop down list named "Active Configuration". Here you can select your board (see Boards and Modules).

Now click "Build" / "Build and Run" or type "Strg+F7" to build and flash the example.

The result files for the AvrStudio build are stored in the directory with the name of the selected configuration, e.g. in this case "rdk230".

Note

Not every example is available for every board, e.g. since the board tiny23x has no UART interface, the examples that requires a HIF are not available.

2.4.3 Atmel Studio 6 Project Template

Introduction

It might of some help, if a preconfigured Atmel Studio 6 template can be used as base for an the own µracoli software project.

Therefore we added the *halimbawa* to the source code package The tagalog word *Halimbawa* means *example* according to the `Tagalog Dictionary`

Using the Template

Step 1: copy the project and rename it.

- Change into the directory `uracoli-src-<version>/as6/`
- Copy the entire directory `halimbawa` and rename it to e.g. `testproject`
- Change to the directory `testproject` and rename the file `halimbawa.cproj` to `testproject.↩ cproj`

Step 2: open the copied project with AS6

- Double click on `testproject.cproj` and wait until Atmel-Studio has started.
- Do a right click on *testproject* in the *solution explorer window* and choose *Properties*.

Step 3: Configure the project

- Find your board on `http://uracoli.nongnu.org/hwlist.html`.
- The yellow marked text is needed to update the AS6 configuration.

- In the properties dialogue select the *Toolchain* tab.
- Adapt the the F_CPU value and enter the *target name* of the board. The preconfigured values are `F_CPU` set to 8MHz and target name `rbb128rfa1`.
- Select the *Device* tab in the *Properties* dialogue.
- Press the *Change Device* button to select the MCU for your board.

Step 4: add your code

Finally you can start with programming your own project. Just edit in the file `main.c`, and leave the other source files untouched ... unless you find a bug.

2.4.4 List of Examples

- Transceiver API
  - xmpl_trx_base.c
  - xmpl_trx_echo.c
  - xmpl_trx_rxaack.c
  - xmpl_trx_rx.c
  - xmpl_trx_txaret.c
  - xmpl_trx_tx.c
- Radio API
  - xmpl_radio_range.c
  - xmpl_radio_stream.c
- HIF API
  - xmpl_hif.c
  - xmpl_hif_echo.c
- Sensor API
  - xmpl_sensor.c
  - xmpl_lm73.c
  - xmpl_isl29020.c
  - xmpl_tsl2550.c
  - xmpl_i2c.c
  - xmpl_ow.c
  - xmpl_lgee_acc_simple.c
- GPIO API
  - xmpl_dbg.c
  - xmpl_key_events.c
  - xmpl_keys.c
  - xmpl_leds.c
- Uilities API
  - xmpl_linbuf_rx.c
  - xmpl_linbuf_tx.c
- Timer/RTC API
  - xmpl_rtc.c
  - xmpl_timer.c
  - xmpl_timer_callback.c

2.5 Applications

2.5.1 Overview

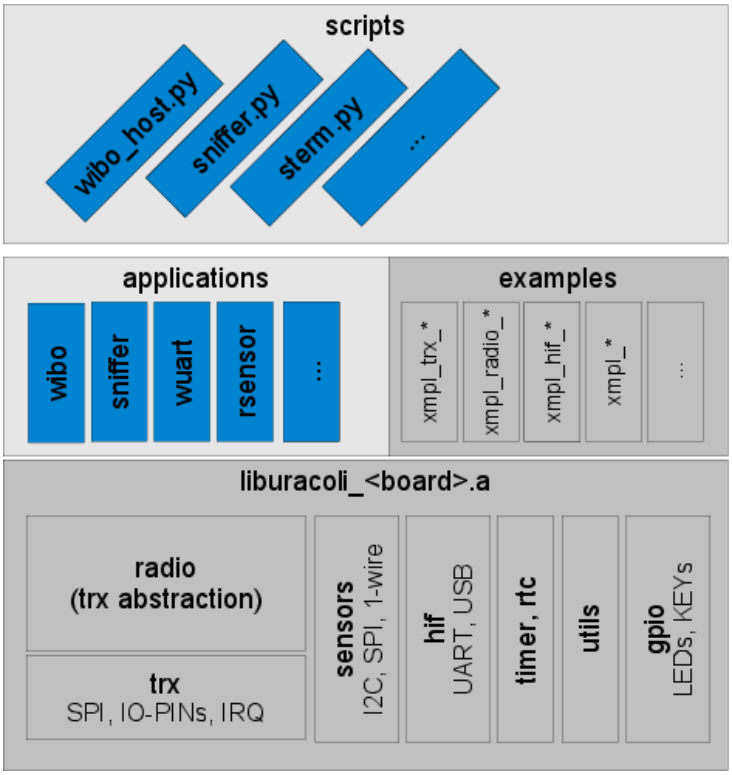


Figure 5: Applications Overview

Here are the applications, made from the code of the µracoli libraries, is described.

- Wireless UART
  The wireless uart can be used for PC to PC connections or as frontend displaying sensor data.
- Radio Diagnostics
  needs rework and concept alignment
- Radio Sensor
  send and receive the board sensor values
- 802.15.4 Packet Sniffer
  Monitor frames of the wirless sensor network using free software.
- Wireless Bootloader
  The wireless bootloader allows firmware updates over the air.
- Arduino IDE support
  A description, how various transceiver boards can be programmed with the Arduino IDE.

Doc Todo

- sterm - A Scriptable Terminal
  A scritpable terminal written in python
- hbm251.py

2.5.2 Wireless UART

wuart - The Wireless UART (needs merge)

2.5.2.1 Overview

The wireless UART can be used to communicate between two PCs via a RF link, or to communicate between a PC and an autnomous sensor/actor node. The WUART application starts on a fixed radio channel and operates automatically in a transparent data mode. With means of the "+++" escape sequence, the menue mode is entered and the application can be configured.

Note

The programm documentation of the firmware can be found in this section.

After a reset of the firmware, the programm prints a startup message and is then in the data transmission mode.

The startup message can be used toidentify the baudrate, that is configured at compile time. The startup message is one line of text and contains the following information:

```
Wuart 0.3 chan=11 radio 83.4 cfg F
      |    |         |        |
      |    |         |        +... configuration read from flash (F) or EEPROM (E)
      |    |         +............ values of transceivers PART_NUM.VERSION_NUM register
      |    +..................... configured IEEE radio channel
      +.......................... application version
```

After printing this startup message the WUART application is in data transmission mode, that means that the payload of incoming frames is printed on the serial interface and incoming data from the serial interface are sent as IEEE 802.15.4 frames.

2.5.2.2 Prerequisites

2.5.2.3 Firmware Installation

2.5.2.4 Configuration

The wireless UART has a configuration menue, that can be reached by typing the Hayes 302 break sequence: `pause +++ pause`. After entering this sequence the following menu will appear in the terminal window:

```
MENU:
[a] node address: 0x5
    peer address: 0x4
[c] channel:      11
[r] reset changes
[e] save and exit
[q] discard chadiscard changes and exit
```

Pressing the letter in the square brackets will query you at a prompt for a new parameter value. Pressing "e" stores the current settings in the EEPROM. After pressing "e" or "q" the application returns to data mode.

2.5.2.5 Frame Protocol

The wireless UART uses the simple P2P air protocol. It assumes IEEE 802.15.4 data frames with 16 bit adressing and PAN-ID compression. The following frame contains just one payload byte (7), the ASCII letter 'c':

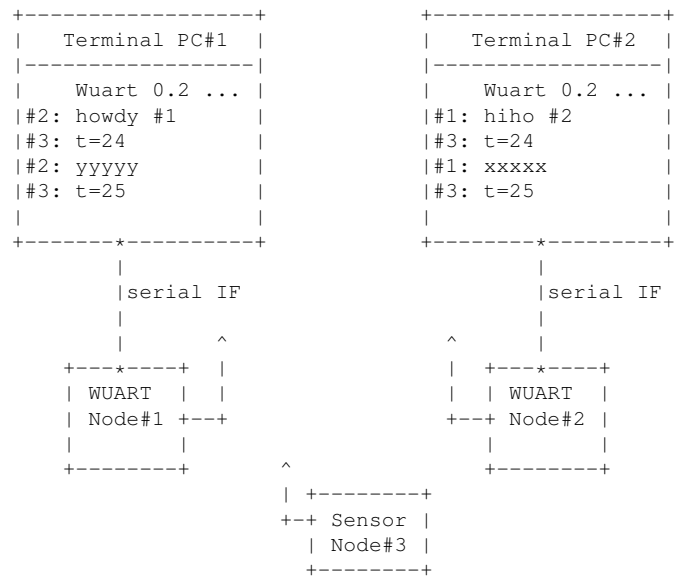
```
Frame Bytes:
0000  41 88 02 42 42 07 00 06 00 40 55 63 e9 b1         A..BB.....
      +---+ ++ +---+ +---+ +---+ +---+ ++ +---+
        (1)  (2) (3)   (4)   (5)   (6)  (7) (8)
```

```
Protocoll Data:
Frame 282: 14 bytes on wire (112 bits), 14 bytes captured (112 bits)
IEEE 802.15.4 Data, Dst: 0x0007, Src: 0x0006
P2P Protocol Data
    FCF:   0x4188               (1)
    Seq:        2               (2)
    pan:   0x4242               (3)
    to:    0x0007               (4)
    from:  0x0006               (5)
    cmd:   P2P_WUART_DATA (64)  (6a)
    mode:  85                   (6b)
    sdata: c                    (7)
    FCS:   0xb1e9               (8)
```

2.5.2.6 wuart - The Wireless UART (needs merge)

Introduction

The wireless UART can be used to communicate between two PCs via a RF link, or to communicate between a PC and an autnomous sensor/actor node. The WUART application starts on a fixed channel and is automatically in the transparent data mode.



Compiling

The wireless UART for the radiofaro board is build with the following commands:

```
cd uracoli-src-<version>
make -C wuart radiofaro
```

The firmware file `bin/wuart_radiofaro.hex` is now available. It can be flashed into the microcontroller e.g. with an AVR-Dragon-programmer using the command:

```
avrdude -Pusb -cdragon_jtag -pm128rfa1 -U bin/wuart_radiofaro.hex
```

Now open a serial terminal programm (e.g. sterm), adjust the baudrate, set the hardware handshake to *none* and after a reset of WUART node you will see a boot message, similiar to this:

```
Wuart 0.2 chan=17 radio 02.01
```

Do the same steps for a second board and try to chat between the terminal windows.

A list of all supported boards is displayed with the command:

```
make -C wuart list
make: Entering directory '/home/axel/Work/uracoli-src-<version>/wuart'
  any2400          : A.N. Solutions ANY Brick
  any2400st        : A.N. Solutions ANY Stick
  ...
  zgbt1281a2uart1  : Meshnetics Zigbit A2, using UART1
  zigduino         : Zigduino made by Logos Electromechanical LLC
```

Alternatively you can compile the example programm `xmpl_linbuf_tx.hex` and watch the text that appears in the terminal window of the PC.

```
make -C xmpl -f xmpl_linbuf_tx.mk <sensorboard>
```

This example can be used as starting point for an own application.

Protocoll

2.5.3 Radio Diagnostics

This application provides a framework for testing the radio functions.

The basic functions of the transceiver, e.g. setting channel, adjusting power and transmission/reception of frames can be done with hotkeys. The application is usefull to check if a new board definition is valid or a hardware platform works as expected.

```
avr-size install/bin/rdiag_rdk230.hex
  text    data     bss     dec     hex filename
     0    7960       0    7960    1f18 install/bin/rdiag_rdk230.hex
```

2.5.3.1 Usage

```
- Load the file @c "install/bin/rdiag_<board>.hex" to your hardware.
- After starting a terminal and connecting the hardware


Radio Diag 0.12
                                                        >RADIO INIT: OK
>RADIO INIT: OK


- Press @b h to display a help screen


i/I : show / show and reset statistic
+/- : incr./decr. channel
P/p : incr./decr. power
C/c : incr./decr. CCA mode
o   : set state OFF
t   : set state TX
r   : set state RX
s   : send a test frame
R   : toggle RXON_IDLE parameter
V/v : incr./decr. verbosity (0...2)


- Press @b i to display a statistic of the radio. <br>
 If the key @b I is pressed, the statistic values are
 reset to 0 after printing.


>STAT duration: 33 ticks
 RX: frames: 12 crcerr: 12
 RX: channel 26 avg rssi: 0 avg lqi:  0
 TX: frames: 0
```

- with the keys **o** (TRX_OFF), **r** (TRX_RX), **t** (TRX_TX), the radio can be set into the basic states

```
>SATE_OFF
>SATE_RX
>SATE_TX
```

- with the keys **+** and **-** the radio channel can be selected (11...26). The initial channel after chip reset is 11 (hardware feature).

```
>CHAN: 12
>CHAN: 11
>CHAN: 26
```

- with **p** and **P** the transmit power is modified. The numbers 0 ... 15 represent the power values given in the data sheet of the AT86RF230
- with **c** and **C** the CCA mode is switched (0...3)

```
>PWR: 9
>PWR: 10
>CCA: 0
>CCA: 3
```

- with **s** a single test frame is send.
- with **S** a test frame is send continously (until **S** is pressed).

```
>SEND len=42
0000  01 00 02 55 55 55 55 55 55 55 55 55 55 55 55 55
0010  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0020  55 55 55 55 55 55 55 55 00 00
>SEND CONTINOUS START
>SEND CONTINOUS STOP
```

- with **R** the transceiver state, while idle is selected
  - 1 after sending a frame, the transceiver is set to state TRX_RX
  - 0 after sending a frame, the transceiver is set to state TRX_OFF

```
>RXON_IDLE: 1
>RXON_IDLE: 0
```

- with **V** and **v** the verbose level can be changed between 0..2
  - at 0 : nothing about TX and RX events is printed
  - at 1 : frame parameters for TX and RX events are printed
  - at 1 : the frame contents is also dumped
- in case of many events, not all messages can be printed and display will miss information

```
>VERBOSE: 0
>VERBOSE: 1
++FRAME len=117, crc=238, lqi=0 rssi=0
++FRAME len=31, crc=192, lqi=0 rssi=0
>SEND len=42
>VERBOSE: 2
++FRAME len=71, crc=119, lqi=0 rssi=0
0000  24 d9 80 79 eb b1 91 cf 6a ab 14 21 f7 f0 ce 6c
0010  6e a6 6c a1 16 37 72 e4 e7 60 56 20 66 27 31 42
0020  41 db 82 66 36 54 88 37 57 37 47 73 1c 7a d7 72
0030  6a 3f 40 82 17 c5 27 24 1d 1e 16 a6 52 71 59 64
0040  66 22 4c 32 71 37 53
>SEND len=42
0000  01 00 03 55 55 55 55 55 55 55 55 55 55 55 55 55
0010  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0020  55 55 55 55 55 55 55 55 00 00
```

2.5.4 Radio Sensor

2.5.4.1 Overview

2.5.4.2 Prerequisites

2.5.4.3 Firmware Installation

2.5.4.4 Terminal Interface

2.5.5 802.15.4 Packet Sniffer

2.5.5.1 Overview

The traffic in a IEEE 802.15.4 network can be monitored with a common transceiver board and the µracoli sniffer firmware. The setup of a sniffer requires the following components:

- A **transceiver board** running the sniffer firmware. It is connected to the PC via UART or USB.
- The Python script **sniffer.py**. It realizes the bridging of the data between the transceiver board and wireshark and it enables the user to select channels and modulations.
- **Wireshark/tshark** as frontend for display and analysis the captured frames.

2.5.5.2 Prerequisites

Additional Software

- Python 2.x with pyserial
- Wireshark
- Firmware Programming Tool

Python with pyserial is used as interface between the host interface (UART or USB) and the Wireshark or tcpdump tool.

Wireshark is a comprehensive and sophisticated network analysis tool.

In order to flash the sniffer firmware to the target board, a flash programming tool (JTAG-ICE, AVR-Dragon, Serial Bootloader, ...) and a programming software is required.

2.5.5.3 Firmware Installation

Compiling

Check out if your hardware supports sniffer functionality

```
$ make -C uracoli-<version>/src/sniffer list
make: Entering directory '/home/uracolix/uracoli-<version>/src/sniffer'

 any2400          : A.N. Solutions ANY Brick
 any2400st        : A.N. Solutions ANY Stick
 any900           : A.N. Solutions ANY Brick
 any900st         : A.N. Solutions ANY Stick
 atrcb256rfr2xpro : Atmel ATmega256RFR2 ZigBit Xplained Pro Extension
 ...
 zgbh230          : ATZGB.com evaluation board
 zgbh231          : ATZGB.com evaluation board
make: Leaving directory  '/home/uracolix/uracoli-<version>/src/sniffer'
```

The firmware is compiled with the command:

```
$ make -C uracoli-<version>/src/sniffer <myboard>
```

and can be found in `uracoli-` <version>/src/bin/sniffer_<myboardh>.hex/elf

Programming

The programming of the firmware and the fuse settings of the μController is explained in the documentantion of your programming software.

Here is an example for flashing a RCB230 on a Sensor Terminal board avrdude via JTAG-ICE or similiar.

```
avrdude -P usb -c jtag2 -p atmega1281 \
    -U lfuse:w:0xe2:m -U hfuse:w:0x11:m -U efuse:w:0xff:m \
    -U sniffer_stb230b.hex
```

According to the transceiver board you want to use as sniffer hardware, the aproriate HEX file and right fuse settings have to be programmed. Here is a list with the settings for the supported plattforms.

Note

If the firmware of your transceiver board is not contained in sniffer package, you can compile the firmware from the µracoli source code package (see http://uracoli.nongnu.org/buildprocess.html).

2.5.5.4 Running the Sniffer

Command Line Usage

The sniffer application is launched with the following command in **Linux**

```
$ python script/sniffer.py -p /dev/ttyUSB0 | wireshark -ki -
```

or with this command in **Windows:**

```
$ python script/sniffer.py -p COM1 | wireshark -ki -
```

All command line options of script are displayed with the command `python script/sniffer.py -h`. The following options are supported

```
-p PORT:
    Serial port, the data rate is optionally seperated by ":", e.g. -p COM1:38400
-c CHANNEL:
    Initial channel to be used.
-r RATE:
    Initial data rate to be used.
-h:
    Show help and exit.
-V:
    Show version and exit.
```

After running this command from the command line, the sniffer control window and the wireshark window will appear on the screen.

Another method of running the sniffer tool is simply to collect the captured data in a logfile:

```
$ python script/sniffer.py -p /dev/ttyUSB0 -c 26 > log.pcap
```

An option of storing the data in the background is using the Linux/Unix command `tee`:

```
$ python script/sniffer.py -p /dev/ttyUSB0 -c 26 | tee log.pcap | wireshark -ki -
```

If just a terminal without an X-Windows is available, the tool `tshark` can be used as backend:

```
$ python script/sniffer.py -p /dev/ttyUSB0 -c 26 | tshark -i -
```

The Lua dissector for the P2P protocoll, that is used, e.g. for the wireless UART, can be enabled with the following command line:

```
$ python script/sniffer.py -p /dev/ttyUSB0 -c 26 | wireshark -Xlua_script:script/p2p.lua -ki -
```

GUI Usage

In the sniffer control window there are two the scroll lists: **Channel** (see option `-c`) and **Rate** (see option `-r`) this transceiver parameters can be canged interactively during a session. With the button **Quit** the GUI and the tool `sniffer.py` is closed, but wireshark remains open. A quick method to exit `sniffer.py` and `wireshark` is pressing "Ctrl-C" at the command line.

2.5.5.5 Terminal Interface

This section describes the UART interface of the sniffer firmware. The interfacing with wireshark is described in section 802.15.4 Packet Sniffer.

The application provides two modes:

- The Scan Mode scans over the selected channels and displays statistics of the received frames. The channel is switched after the time SCAN_PERIOD_MS.
- The Sniff Mode receives on one selected channel and dumps the frames on the host interface. This mode can used for live capturing WPAN data with wireshark (see contibution guide for more details).

Usage

The following commands are available in the terminal:

Mode Commands:

- ***idle***, Keys: `I`, `<SPACE>`
  Set application to idle mode and transceiver to state TRX_OFF.
- ***scan***
  Scan for frames on the enabled channels (see also commands `cmask`, `cmset` and `cmclr`)
- ***sniff***
  Start sniffing frames on the current channel and dump the received packets of the hostinterface (see command `parms`).

Parameter Commands:

- ***parms***
  Display current parameter settings.
- ***chan*** <***CHAN***>, Keys: + , −
  Set current channel (for sniffing). "+" increments the current channel by one, "-" decrements by one.
- ***cmclr*** <***chan***>
  Erase the channel bit in `cmask`.
- ***cmset*** <***chan***>
  Set the channel bit in `cmask`.

- ***cmask*** <***mask***>
  Set the 32 bit channel mask. `mask` can be a hexadecimal value, e.g. 0x07fff800
- ***cpage*** <***page***>
  Set current channel page ({not yet supported})
- ***drate*** <***ratestr***>
  Set current data rate of transceiver. ratestr = {BPSK{20,40}, OQPSK{100,200,250,400,500,1000,2000}}
- ***chkcrc*** <***[0|1]***>
  if 0, discard frames with invalid CRC, if 1, report also frames with invalid CRC.

Scan Mode

The Scan Mode is started with the command `scan`. The scanned channels can be configured with the channel mask and the commands `cmset`, `cmclr` and `cmask`.

IEEE Channels, Channel Mask and Channel Page in Scan Mode

IEEE 802.15.4 uses a 32 bit channel mask. Since the standard uses just 26 channels (0 for 868 MHz, 1-10 for 915 MHz and 11-26 for 2.4 GHz) the upper 5 bits of the channel mask are currently used by the value channel page. The channel page identifies the modulation method used.

In order to enable channel 15 in the channel mask for scanning, you can use the command `cmask 0x00008000` or `cmset 15`.

```
> parms
SUPP_CMSK: 0x07fff800 ....... shows the supported channels of the radio
CURR_CMSK: 0x04008800 ....... channel mask used for scanning (11,15,26)
CURR_CHAN: 11 ............... channel used for sniffing
CURR_PAGE: 0  ............... unsupported yet
```

Display of the Scan Results

In scan mode the application switches from channel to channel and collects the statistics of the received frames. The information can be displayed in a terminal as table (see example below, which scans only channels 11 and 15, cmask = 0x00008800).

```
scan


Scanning channels, scan period 2000ms
             bad    Avg.          802.15.4 frames
chan   frm   crc   ed lqi      B     D     A     C  PER
 11      0     0    0   0      0     0     0     0    0
 12  n/a
 13  n/a
 14  n/a
 15     62     0   71 255      0    62     0     0    0
 16  n/a
 17  n/a
 18  n/a
 19  n/a
```

```
 20  n/a
 21  n/a
 22  n/a
 23  n/a
 24  n/a
 25  n/a
 26      1     1    0   0      0     0     0     0  100
=== ur 0 err 0 frames: 63 ===
```

The columns of the table have the following meaning:

```
             bad    Avg.          802.15.4 frames
chan   frm   crc   ed lqi      B     D     A     C  PER
 |      |     |    |   |       |     |     |     |   +... Estimated packet error rate
 |      |     |    |   |       |     |     |     +....... number of rx'd command frames
 |      |     |    |   |       |     |     +............. number of rx'd ACK frames
 |      |     |    |   |       |     |     +............. number of rx'd data frames
 |      |     |    |   |       |     +....................... number of rx'd beacon frames
 |      |     |    |   |       +........................... number of rx'd beacon frames
 |      |     |    |   +................................... average LQI of frames on this channel
 |      |     |    +....................................... average energy value frames on this chan
 |      |     +......................................... number of frames with bad CRC
 |      +............................................... number of received frames
 +..................................................... channel number
```

Sniffer Mode

The Sniff Mode is started with the command `sniff`. The channel which should be monitored is set with the command `chan`. In the sniff mode, each received frame is transfered over the host interface to the PC application.

Data Transfer in Sniffer Mode

The binary protocoll used for transfer the received frames over the host interface in sniff-mode has the following format:

```
+-------+--------+--------+------------+--------+
| SOF   |  LEN   | TSTAMP | FRAMEDATA  | EOF    |
+-------+--------+--------+------------+--------+
| 1Byte | 1 Byte | 8 Byte | 1-127 Byte | 1 Byte |
+-------+--------+--------+------------+--------+
```

Parameters

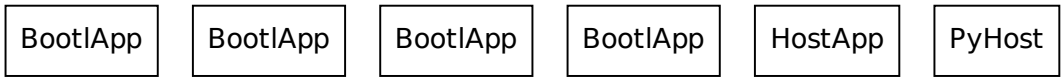
SOF	Start of frame field (0x01)
LEN	Length of packet (size of time stamp and frame size) (8 ... 133)
TSTAMP	Time structure (4 Byte seconds, 4 byte micro seconds)
FRAMEDATA	Octets of the frame (the data does not contain the IEEE 802.15.4 PHR with the reserved bit value).
EOF	End of frame field.

2.5.6 Wireless Bootloader

This application implements a wireless Bootloader

The wireless bootloader WiBo consists of three main parts

- The bootloader app itself, residing in the bootloader section of the remote nodes
- The host application running on board connected to a PC, this can be different from the target remote boards
- A python class that implements all functions required for flashing firmware



2.5.6.1 Overview

The following article describes the usage of the uracoli wireless bootloader. WiBo works like a regular bootloader, except that it uses the radio transceiver instead of a UART. WiBo modifies the flash directly, therefore no special backup flash memory, like in other over the air upgrade (OTA) solutions, is required on the transceiver board.

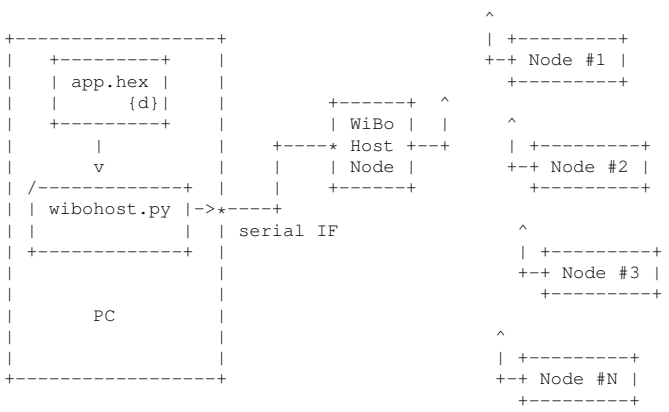
2.5.6.2 Prerequisites

The WiBo framework provides a method to program the MCU that controls the transceiver. Three components are involved,

- a PC with USB or serial interface, running the script wibohost.py,
- a host node, running with the wibohost firmware and
- the network nodes that have the bootloader installed.

The python script *wibohost.py* sends a hexfile *app.hex* in slices to the host node. It transmits these data slices as unicast or broadcast frames to the network nodes. The network nodes collect the data slices and programm them with SPM (self programming mode) commands in the flash application section of the microcontroller.

In order to verfify if a node was programmed correctly, the host and network nodes calculate a CRC16 during the transmission and reception of the data slices. When programming is completed, the wibohost can query the CRC from the nodes and compare with its own checksum.



The bootloading can be done either in *unicast* mode or in *broadcast* mode. In unicast mode one node is programmed with a program image at one time. In broadcast mode, multiple nodes receive the image in parallel. This is useful for programming multiple instances of identical hardware with a firmware image.

The host site

consists of a Python script and the host node that runs the wibohost firmware. The script *wibohost.py* uses the module pyserial for serial communication with the host node. It reads and parses the given hex file and transfers it in slices to the host node. The host node firmware sends this slices in frames to one (unicast) or all (broadcast) network nodes.

The wireless bootloader

resides in bootloader section of the network nodes and occupy about 1K words of programm memory. Additionally to the bootloader code, configuration record at the end of the bootloader section. This record ensures that the node address and channel information is available, even if the EEPROM was accidentely erased.

The network nodes are passive, that means they never send anything if not queried by the host node.

2.5.6.3 Node Configuration

Before the Atmel radio transceivers can be used in a wireless network scenario, they need to be configured with some key parameters.

The most important parameter is the radio channel, e.g. the frequency on which the radios communicate. Depending of the radio type, the cannel can be 11 - 26 for the 2.4GHz radios and 0 - 10 for the 868/900 MHz radios. Which channel to choose depends on the area where the network is operated (for 868/900 MHz) and which other interferers are present (e.g. Bluetooth and WLAN for 2.4GHz). Since this topic can't be covered here complete, consult other information sources for selecting a radio channel to operate on.

Other important parameters are the node address and the network address. The IEEE-802.15.4 radios support a 16 bit (SHORT-ADDRESS) and a 64 bit (IEEE-ADDRESS) node address as well as a 16 bit network identifier (PAN-ID).

Since for bootloaders with the limited amount of programm memory no complex search and asociation procedures can be implemented, the channel and addressing parameters needs to be stored persistently in the Flash or EE↩ PROM memory of the network node.

Since the EEPROM in AVR controllers can be easily erased or modified by the application, a safe place to store the radio configuration is the flash memory (the programm memory itself). Since the parameters should be accessible from the bootloader and from the final application, a so called configuration record is stored at the end of the flash memory.

In order to create the config record at the flash end, a Intel Hex file needs to be generated, that is flashed with the hardware programmer and .e.g. with the tool avrdude.

For generating config records for multiple nodes, the pythonscript +wibo/nodeaddr.py+ is used. Basically the tool can be used directly on the command line.

```
$python wibo/nodeaddr.py -B rdk230 -c 17 -a 1 -A 0x1234567890ABCDEF -p 0xcafe -o foo.hex
Use Parameters:
 board:      rdk230
 addr_key:   1
 group_key:  None
 short_addr: 0x0001
 pan_id:     0xcafe
 ieee_addr:  0x1234567890abcdef
 channel:    17
 offset:     0x0001fff0
 infile:     None
 outfile:    foo.hex
```

More detailed information about the tools options are displayed with the command

```
$python wibo/nodeaddr.py -h
```

As you see in the above example a command line that covers all relevant parameters is very long and therefore error prone too. The script +nodeaddr.py+ can read its parameters also from a configuration file.

A initial nnotated config file for a network setup is created with the command:

```
$python wibo/nodeaddr.py -G mynetwork.cfg
generated file mynetwork.cfg
$cat mynetwork.cfg
# In the "groups" section several nodes can be configured to be in one group,
# e.g. 1,3,4,5 are ravenboards.
#[group]
#ravengang=1,3:5
....
```

Assuming that your network consists just of rdk230 boards, the file +mynetwork.cfg+ for the above command line would look so:

```
[board]
default = rdk230

[channel]
default=17

[pan_id]
default=42

[ieee_addr]
0=0x1234567890abcdef

[firmware]
outfile = node_<saddr>.hex
```

The HEX files for the nodes are generated with the next command:

```
$python wibo/nodeaddr.py -C mynetwork.cfg -a 0
$python wibo/nodeaddr.py -C mynetwork.cfg -a 1
```

or in very freaky pipeline way:

```
$python wibo/nodeaddr.py -Cmynetwork.cfg -a0 -o- | avrdude <OPTIONS> -U fl:w:-:i
```

2.5.6.4 The Host Application

Compiling and Flashing

Here is an example for the Raven USB Stick. Applying the configuration record to the hex-file is done in the same manner as for the bootloader application.

```
    cd wibo
make -C ../src rzusb
make -f wibohost.mk rzusb
python nodeaddr.py -a 0 -p 1 -c 11 -f ../bin/wibohost_rzusb.hex -B rzusb -o h.hex
avrdude -P usb -c jtag2 -p at90usb1287 -U h.hex
```

Note

The option "-p 1" set the IEEE PAN_ID to "1" and must be identical for the bootloader application and the wibohost application.

2.5.6.5 Python Host Application

Using wibohost.py

To test the wireless bootloader environment, the xmpl_wibo application will be used. It blinks a LED with a certain frequency and is able to jump in the bootloader when the special "jump_to_bootloader" frame is received.

At first create some firmware versions, e.g. one slow and one fast blinking. The network nodes shall be rdk230 nodes.

```
make -f xmpl_wibo.mk BLINK=0x7fffUL TARGET=slow rdk230
make -f xmpl_wibo.mk BLINK=0xffffUL TARGET=fast rdk230
```

Next assume that you have 4 network nodes with addresses [1,2,3,4]. In order to check the presence of the nodes, run the scan command.

```
python wibohost.py -P COM1 -S
```

Note that the default address range of wibohost.py is 1 ... 8. This can be modified with the -A option. In the example above, only the nodes 1 to 4 are present, therefore no response from the nodes 5 ... 8 is received.

At first we update all nodes with the slow blinking firmware. Therefore we use the broadcast mode (-U), that means that the image is transfered only once over the air. The address range (-A) is needed to ping the nodes before programming and afterwards to verify their CRC.

```
python wibhost.py -P COM1 -A1:4 -U slow.hex
```

In the next step we selectively flash node 1 and node 3 with the file fast.hex. Since we use unicast programming (-u), the image is transfered for each node over the air seperately.

```
python wibhost.py -P COM1 -A1,3 -u fast.hex
```

2.5.6.6 The Bootloader Application

Build and Flash

Build the bootloader for your board with the command

```
make -C ../src <board>
make -f wibo.mk <board>
```

With the command +make -f wibo.mk list+ the available <board>s are displayed.

The bootloader expects an address record at the end of the flash memory section. This record can be generated with the script nodeaddr.py. Here is an example for the rdk230 board.

```
# generate a hex file with the configuration record for node #1
python nodeaddr.py -a 1 -p 1 -c 11 \
       -f ../bin/wibo_rdk230.hex -B rdk230 -o a1.hex

# flash node #1 (SHORT_ADDR=1)
avrdude -P usb -p m1281 -c jtag2 -U a1.hex

# Fuses for initial jump to bootloader
avrdude -P usb -p m1281 -c jtag2 -U lf:w:0xe2:m \
        -U hf:w:0x98:m -U ef:w:0xff:m
```

To verify the correct AVR fuse settings refer to http://www.engbedded.com/fusecalc.

To flash multiple nodes more efficiently, the nodeaddr.py can pipe its output directly into avrdude. This slightly more complex command line can be stored in script +flashwibo.sh+ (under Windows replace $1 by %1 for +flashwibo.bat+):

```
python nodeaddr.py -a $1 -p 1 -c 11 -f ../bin/wibo_rdk230.hex -B rdk230 |\
       avrdude -P usb -p m1281 -c jtag2 \
                -U fl:w:-:i -U lf:w:0xe2:m -U hf:w:0x98:m -U ef:w:0xff:m
```

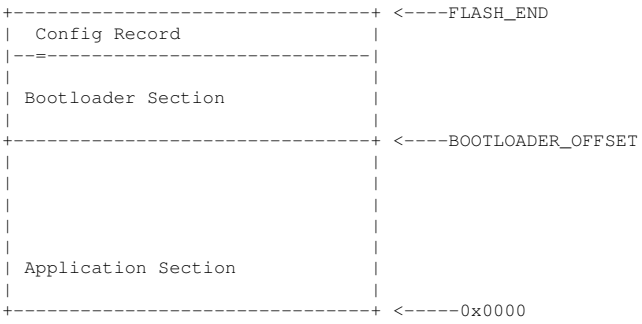
Note

The option "-p 1" set the IEEE PAN_ID to "1" and must be identical for the bootloader application and the wibohost application.

With the +python nodeaddr.py -h+ the help screen is displayed.

Flash memory partitioning

The AVR flash memory can be divided in an application and a booloader section. The application section is located in the lower address memory. The bootloader section is located in the upper flash memory. In this section the so called self programming opcodes (SPM) can be executed by the AVR core. This SPM opcodes allows erasing and reprogramming the application flash memory.



The start address of the bootloader section is determined by the BOOTLSZ fuse bits. The BOOTRST fuse bit determines, if the AVR core jumps after reset to the application section (address 0x0000) or to the bootloader section (e.g. address 0xf000). The AVR fuse bits and the content of the bootloader section can only be changed either by ISP, JTAG or High Voltage programming.

In order to have enough memory for the application available, the booloader section is choosen to be rather small, e.g. for 8K devices a 1K bootloader section and 7K application section is a reasonable choice. The larger 128k AVR devices are partitioned usually with a 4K bootloader section, leaving 124K flash memory for the application.

The Configuration Record at FLASH_END

For WiBo, the last 16 byte of the flash memory are reserved for a configuration record, that holds address and channel parameters, which are needed for operation. The structure of this record is defined in file +board.h+ in the type +node_config_t+. It stores

- 2 byte SHORT_ADDRESS
- 2 byte PAN ID
- 8 byte IEEE_ADDRESS
- 1 byte channel hint
- 2 reserved bytes
- 2 byte CRC16

The configuration record is accessible from the applicatation and from the bootloader section.

2.5.6.7 The WiBoHost API

The file wibohost.py can also be used as a python module. The following script shows how a broadcast and a unicast flash can be programmed.

```
from wibohost import WIBOHost

wh = WIBOHost()
# open serial connection to wibohost node
wh.close()
wh.setPort("COM19")
wh.setBaudrate(38400)
wh.setTimeout(1)
wh.open()
# check if local echo works.
print WHOST.echo("The quick brown fox jumps")

# scan addresses 1 to 4
addr_lst = wh.scan(range(1,4+1))

# broadcast mode, flash all nodes
for n in addr_lst:
    wh.xmpljbootl(n)
    print "PING  :", n, wh.ping(n)

print "FLASH :", wh.flashhex(0xffff, "foo.hex")
for n in addr_lst:
    print "CRC  :", n, wh.checkcrc(n)
    print "EXIT :", n, wh.exit(n)

# unicast mode, flash node 1
wh.xmpljbootl(1)
print "PING  :#1", wh.ping(1)
print "FLASH :#1", wh.flashhex(1, "bar.hex")
print "CRC   :#1", wh.checkcrc(1)
print "EXIT  :#1", wh.exit(1)
```

2.5.7 Arduino IDE support

2.5.7.1 Overview

µracoli provides a support package for the Arduino-IDE. The UASP (µracoli-arduino-support-package) includes a Arduino-Serial-like library for the radio transceiver and board definitions for some selected boards.

Arduino is a Microcontroller Development plattform that consists of a Java-IDE which supports various microcontroller boards. Most of the supported boards are equipped mit 8-bit-AVR microcontrollers.

Basically the Arduino-IDE provides a simple code editor and the firmware (denoted as "sketches") can be compiled and flashed with a button click. A serial terminal completes the IDE. This high level of abstration makes it very easy for none technicians and first-time users to start with embedded programming.

A **sketch** basically implements two functions `setup()` and `loop()` that are called from the main function of the core library. A simple API is provided, that is described at http://arduino.cc/en/Reference/HomePage.

µracoli provides a support package package for the Arduino-IDE with versions above 1.5.x.

The following boards are supported in the package:

- radiofaro - Radiofaro w/ ATmega128rfa1
- pinoccio - Pinoccio Scout
- atzb256rfr2xpro - Atmel ATmega256RFR2 Xplained Pro Evaluation Kit
- wdba1281 - Zigbit 2400MHz, w/ ATmega1281
- mnb900 - Zigbit 900MHz, w/ ATmega1281

The Arduino project did fork in 2014, see http://hackaday.com/2015/04/06/arduino-ide-forked/ for details. Since this time there are two versions of the IDE avaiable.

- IDE from arduino.cc
  https://www.arduino.cc/en/Main/Software
- IDE from arduio.org
  http://www.arduino.org/downloads

Note

In version arduino.cc-v1.6.5/arduino.org-v1.7.3 the arduino.org IDE comes with a complete arm-none-eabi GNU toolchain and a CMSIS for Atmel Cortex-M0/M0+/M4 controllers. The arduino.cc IDE is the SAM and SAMD board variants are missing too. The rest of both IDE-packages seems to be identical.

2.5.7.2 Installation

Download and install one of the Arduino IDEs.

Select one of the forked IDEs

- IDE from arduino.cc
  https://www.arduino.cc/en/Main/Software
- IDE from arduio.org
  http://www.arduino.org/downloads

and install it on your computer according to the instructions for your OS (Linux, Windows, MAC-OS).

Download the µracoli-Arduino-Support-Package (UASP)

Download the UASP-zipfile `uracoli-arduino-15x-` <version>.zip from http://uracoli.nongnu.org/download.html

Option A: Install UASP in the IDE directory

With this method the package is installed centrally for all users. The UASP example sketches are located in "`File / Examples / Radio`"

- Change to the Arduino-IDE directory (e.g. "`cd /opt/arduino-1.6.5/`")
- Unpack the UASP (e.g. "`unzip uracoli-arduino-15x-<version>.zip`")
  The UASP has three top level directories, `hardware`, `examples`, `doc`) the fell directly in the existing Arduino-IDE directories.
- Open the Arduino-IDE (e.g. "`./arduino`").
- If you see in the menu "`Tools / Board`" the boards Radiofaro, Zigbit2400 and Zigbit900, the installation was successful.

Option B: Install UASP in the Sketchbook folder

This method installs the UASP locally in the sketchbook folder of the current user. The UASP example sketches are located in "`File / Sketchbook / Radio`"

- Open the Arduino IDE (e.g. "/opt/arduino-1.6.5/arduino")
- Open the menu item "Files / Preferences")
  a dialogue window pops up. Here you find the location of the "Sketchbook location" in the first entry.
- Change to the "Sketchbook location", e.g. "cd /home/axel/Arduino".
- Unpack the UASP (e.g. "unzip uracoli-arduino-15x-<version>.zip")
  The UASP has three top level directories, hardware, examples, doc) the fell directly in the existing Arduino-IDE directories.
- Reopen the Arduino-IDE (e.g. "/opt/arduino-1.6.5/arduino") to load the UASP boards and examples into the IDE.
- If you see in the menu "Tools / Board" the boards Radiofaro, Zigbit2400 and Zigbit900, the installation was successful.

2.5.7.3 Usage

Using the HelloRadio sketch

Restart the Arduino-IDE after unpacking uracoli-arduino- <version>.zip and check if you see the new boards at the end of the list that opens if you click the menu entry "Tools/Board" and select your radio board, e.g. "Radiofaro", "Zigbit 2400MHz", etc.

In the next step select the in the menu "Tools / Serial Port" the serial port to which your radio board is connected.

Now open the HelloRadio sketch. It can be found either in "File / Examples / Radio" or "File / Sketchbook / Radio", depending on the installation option you did choose (see Installation).

Now select the menu entry "File / Upload". The sketch will compiled and flashed to the choosen radio board.

Note

If the upload fails, mark the options "compile verbose" and "upload verbose" in the dialogue that opens when clicking "File / Preferences" and check in the lower window pane of the Arduino-IDE what goes wrong.

The sketch "HelloRadio", that you have currently uploaded, sends a short frame every 500ms on channel 17 and reports the transmission also on its serial port. You can open a terminal with "Tools / SerialMonitor" and you should see in the terminal window:

```
HelloRadio
TX: 0
TX: 1
TX: 2
...
```

Each printed number shows, that a frame was successfully transmitted. If you see this output, that means that you are now "on air".

Example Sketches

A good starting point for using the radio functions are the example sketches.

- IoCheck.ino
- RadioUart.ino

- HelloRadio.ino
- IoRadio.ino
- Gateway.ino
- Remote.ino

Function Reference

The regular Arduino core functions are documented at http://arduino.cc/en/Reference/HomePage.

The radio specific functions are described in section Arduino Radio Functions.

Building Sketches from Command Line

Beside several thirdparty CLI tools, e.g. Arscons, Inotool, since version 1.5 the Arduino IDE supports its own CLI, see https://github.com/arduino/Arduino/blob/ide-1.5.x/build/shared/manpage.adoc

```
arduino --verify \
        --board uracoli:avr:radiofaro \
        examples/Radio/RadioUart/RadioUart.ino
```

Note

- In order to explicitely set the Arduino build path add -pref build.path=<your_build_dir> to the command.
- With option -v, a more verbose output of the build process is generated and the current Arduino build directory is shown.

2.5.7.4 Bootloader

If the Arduino Bootloader in your board is accidently erased, you can restore it with the following procedure.

In order to flash the bootloader, use a flash programmer and a programming tool of your choice (avrdude, atprogram, Atmel-Studio) and flash the file to the board.

The source code and a precompiled Intel-Hex file of the bootloader is located in the directory hardware/uracoli/avr/bootloa

Check also if the fuses are set correctly:

```
LF = 0xe7
HF = 0x90
EF = 0xf6
```

Examples

The preferred flashing program is avrdude since it comes with the Arduino distribution.

Note

: If you use avrdude from the arduino package, an error message about the missing config file may occur.

```
avrdude: can't open config file "...avrdude.conf": No such file or directory
avrdude: error reading system wide configuration file
```

So you have to give the path to the config file explicitely.

Examples

```
# Connection AVR dragon via isp
PROG=avrdragon_isp
# Connection AVR dragon via jtag
PROG=avrdragon_jtag

avrdude -C $ARDUIONDIR/hardware/tools/avr/etc/avrdude.conf \
        -P usb -c $PROG -p m128rfa1 \
        -U lf:w:0xe7:m -U hf:w:0x90:m -U ef:w:0xf7:m \
        -U fl:w:ATmegaBOOT_radiofaro.hex
```

If the bootloader is flashed correctly, it can be checked with

```
avrdude   -C $ARDUIONDIR/hardware/tools/avr/etc/avrdude.conf \
          -P <MYSERIALPORT> \
          -b 57600 -c arduino -p m128rfa1 -U <MYHEXFILE>
```

2.5.7.5 Licenses

This package incorporates source code from different license models, which has an influence on the usae of the code in proprieatary projects and environments.

GPL version 2.0

According to the file header, the bootloader is licensed under GNU General Public License version 2.0. See http://www.gnu.org/licenses/gpl-2.0.txt or file link:license_gpl_2v0.txt[].

```
hardware/uracoli/bootloaders/radiofaro/ATmegaBOOT.c
```

LGPL version 2.1

The files copied from the original Arduino core are licensed under the GNU Lesser General Public License version 2.1. This code is linked to each sketch. See http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt or file link:license_lgpl_2v1.txt[].

```
hardware/uracoli/variants/zigbit900/pins_arduino.h
hardware/uracoli/variants/zigbit2400/pins_arduino.h
hardware/uracoli/variants/radiofaro/pins_arduino.h
```

Modified BSD license

The sources of the µracoli radio functions are licensed under the modified 3 clause BSD license. See file link:license_uracoli.txt[].

```
hardware/uracoli/cores/uracoli/trx_rf230_param.c
hardware/uracoli/cores/uracoli/const.h
hardware/uracoli/cores/uracoli/boards/base_zdma1281.h
hardware/uracoli/cores/uracoli/boards/board_derfa.h
hardware/uracoli/cores/uracoli/boards/board_wdba1281.h
hardware/uracoli/cores/uracoli/trx_rf230_sram.c
hardware/uracoli/cores/uracoli/trx_datarate_str.c
hardware/uracoli/cores/uracoli/board.h
hardware/uracoli/cores/uracoli/at86rf230b.h
hardware/uracoli/cores/uracoli/trx_rf230.c
hardware/uracoli/cores/uracoli/at86rf212.h
hardware/uracoli/cores/uracoli/atmega_rfa1.h
hardware/uracoli/cores/uracoli/trx_datarate.c
hardware/uracoli/cores/uracoli/trx_rf230_frame.c
hardware/uracoli/cores/uracoli/trx_rfa.c
hardware/uracoli/cores/uracoli/trx_rf230_irq.c
```

```
hardware/uracoli/cores/uracoli/trx_rf230_bitwr.c
hardware/uracoli/cores/uracoli/trx_rf230_bitrd.c
hardware/uracoli/cores/uracoli/radio_rf230.c
hardware/uracoli/cores/uracoli/at86rf230a.h
hardware/uracoli/cores/uracoli/radio_rfa.c
hardware/uracoli/cores/uracoli/radio.h
hardware/uracoli/cores/uracoli/trx_rf230_crc.c
hardware/uracoli/cores/uracoli/transceiver.h
hardware/uracoli/cores/uracoli/usr_radio_irq.c
hardware/uracoli/cores/uracoli/trx_rf230_misc.c
examples/Radio/RadioUart/RadioUart.ino
examples/Radio/IoCheck/IoCheck.ino
examples/Radio/HelloRadio/HelloRadio.ino
examples/Radio/IoRadio/IoRadio.ino
```

2.5.8 sterm - A Scriptable Terminal

- Author: Axel Wachtler
- Date: 2010/07 - 2015/03

Abstract

While developping and debugging wireless applications, it is usefull to connect two or more nodes via a serial interface with a PC in order to watch and analyze the debugging output. The programm sterm.py is (yet another) terminal application written in Python/TKinter, that provides this multiport feature.

Prerequisites

In order to run `sterm.py` an installed version of http://www.python.orgPython and the module `pyserial` is needed. A short decription how to install and upgrade Python on your PC can be found under http://uracoli.nongnu.org/gettingstarted.html#_installing_and_upgrading_↩python[Installing and Upgrading Python]. If you will use a transceiver board with an USB interface, of course the correct USB driver must be loaded.

Running sterm.py

The program `wuart/sterm.py` is simply started by the command:

```
python sterm.py
```

If the configuration file `sterm.cfg` is not found in the current directory, a default version of this file is created and must be modified, to reflect your configuration.

With the command `python sterm.py -h` a list of available command line options is displayed.

```
Usage:
 python sterm.py [OPTIONS]

Options:
 -h, --help
    show this help message
 -v, --version
    show version number
 -c FILE, --config FILE
    load config file, default value `sterm.cfg`
    if FILE is not existing, an annotated version
    is generated and the tool is exited.
```

Configuring sterm.py

Overview

The setup is configured with the file `sterm.cfg`. This file is written in WIN.INI format and consists of the following sections.

`[sterm.py]`:: The section name "CONFIG" is reserved for the global configuration of `sterm.py`. `[mydevice]`:: A device section defines the serial port settings and other configuration parameters. It can have a arbitary but unique (within `sterm.cfg`) name. `[mymacro]`:: A macros section defines either a text or a smart macro. It can have a arbitary but unique (within `sterm.cfg`) name.

The `[sterm.py]` Section

The CONFIG section holds the global configuration information.

```
[sterm.py]
devices = .... list of enabled devices
plugins = .... list of files with smart macro functions
macros  = .... macros enable globally for all devices
```

The `[Device]` Section

The example file `sterm.cfg` defines three devices: coord, dev1, dev2, dev3. Each devices is described within a configuration section, which has the following contents:

```
[coord]
type = serial
port = /dev/ttyS4
baudrate = 9600
logname = coord.log
logmode = w
echo = 1
log = 1
connect = 1
macros = mycommand_01 mycommand_02
```

The entries have the following meaning:

- type: currently only value "serial" is usefull.
- port: the name of the serial port, under Windows probably COMx
- baudrate: the serial baudrate to be used (in bit/s, e.g. 9600, 19200, 57600)
- logname: name of the generated logfile
- logmode: "a" for append to the file or "w" to write file from start
- log: a 1 means that writing to the logfile is enabled right at startup, a 0 means that you need to click the "log" button to open the logfile.
- echo: 1 local terminal echo is enabled at startup
- connect: 1 means that the serial connection is opened at programm startup
- macros: list of macros for this device.

After defining a device, it needs to be listed in the "[CONFIG]" section in the keyword "devices", e.g.

```
[sterm.py]
devices = coord dev1 dev2 dev3
....
```

The [Macro] Section

The program sterm.py provides two types of macros: a) normal text macros and b) smart macros, which are implemented by python functions.

Here are two examples for a text macros:

```
[qbf]
text = The quick brown fox jumps over the lazy dog.

[peli]
text = A wonderful bird is the pelican,
        His bill will hold more than his belican,
        He can take in his beak
        Enough food for a week
        But I'm damned if I see how the helican!
```

The "qbf" macro sends a single line to the serial device and the "peli" macro sends multiple lines to the serial device.

Smart macros are configured by the keywords "function" and "params". The "function" keyword holds the name of a function that is defined in a plugin file. The "params" keyword is a python expression that creates a parameter dictionary the function is called with.

```
[mycommand_01]
function = mycommand
params = dict(nbdevices = 1, channel = 0, chanpg=5, test=3)
```

Assigning Macros

The macros are buried behind the button "Macros" as a drop down list. The macro assignement can be global in the "[CONFIG]" section or local in the "[mydevice]" device specification section.

Here are two assignment examples:

```
[sterm.py]
devices = coord dev1 dev2 dev3
plugins = plugin.py
macros = qbf

[mydevice]
type = serial
....
macros = mycommand_01
```

The macro "qbf" is assigned globally to all defined devices, where the macro "mycommand_01" is just assinged to the device "mydevice".

Using Smart Macros

Writing Smart Macros

Smart Macros are implemented by a python function. Here is an example:

```
00001 import time
00002 def mycommand(devices, nbdevices, test, channel, chanpg, pan = 1):
00003     print "test=%d, channel=%d, chanpg=%d, test=%d, pan=%d" %  \
00004           (test, channel, chanpg, test, pan)
00005     cmd = "%02d%02d%1d%02d" % (channel, chanpg, pan, test)
00006     print "Run: %s" % cmd
00007     devices['coord'].write("Sc%s" % cmd)
00008     if nbdevices >=1:
00009         time.sleep(1)
00010         devices['dev1'].write("S1%s" % cmd)
00011     if nbdevices >=2:
00012         time.sleep(1)
00013         devices['dev2'].write("S2%s" % cmd)
00014     if nbdevices >=3:
00015         time.sleep(1)
00016         devices['dev3'].write("S3%s" % cmd)
```

The first parameter "devices" of the function `mycommand()` holds a dictionary of all defined serial devices. By means of the device function "write" the smart macro function can send data to all available devices. All other

parameters (e.g. nbdevices, test, channel, chanpg, pan = 1) are function specific and given by the `params` keyword in the configuration file.

Loading Smart Macros

Smart macros are loaded in the "[sterm.py]" section of the config file just by writing the file name (with a path component) behind the keywords "plugins".

```
[sterm.py]
...
plugins = plugin.py
...
```

2.6 HOWTOs

This part of the µracoli manual gives information about the build process, the tools are used, how to add new hardware to the project and explains how to deal with the USB drivers for some of the boards.

2.6.1 secUSBDevs

- Software
  - Tool Chain and Build Process
  - Openocd self compiling on Ubuntu
- Hardware
  - Programming and Debugging Targets
  - How to add a new radio board to the library
- Misc
  - USB drivers
  - Random Notes

2.6.2 Tool Chain and Build Process

The tool chain needed for compiling µracoli can be setup under Linux, ∗BSD and Windows.

Basically the following tools are needed.

- avr-gcc and avr-binutils,
- a build system tool:
  - gnu-make if the uracoli-src∗.zip package is used.
  - python and scons if the uracoli-devel∗.zip package is used.
- an programming tool, like avrdude or AvrStudio
- optionally doxygen and graphviz if this documentation shall be generated.

A detailed description of the tool chain setup can be found at http://uracoli.nongnu.org/gettingstarted.↩html

2.6.3 Openocd self compiling on Ubuntu

Links

- https://github.com/RIOT-OS/RIOT/wiki/OpenOCD
- http://jjmilburn.github.io/2014/09/18/Atmel-SAMD20-EclipseCDT/
- http://karibe.co.ke/2013/08/setting-up-linux-opensource-build-and-debug-tools-for-fr

Commands

```
dmesg
sudo apt-get install build-essential autoconf automake libtool libusb-dev libusb-1.0-0-dev libhidapi-dev
sudo apt-get install build-essential autoconf automake libtool libusb-dev libusb-1.0-0-dev
sudo apt-get install libudev-dev
cd ..
git clone git://github.com/signal11/hidapi.git
mv hidapi hidapi-git-anon
cd hidapi-git-anon/
./bootstrap
./configure
make
sudo make  install
wget http://sourceforge.net/projects/openocd/files/openocd/0.9.0/openocd-0.9.0.zip
unzip  openocd-0.9.0.zip
cd openocd-0.9.0/
./configure --prefix=/opt/openocd-0.9.0
make
sudo make install
/opt/openocd-0.9.0/bin/openocd
sudo ln -s /usr/local/lib/libhidapi-hidraw.so.0 /usr/lib/libhidapi-hidraw.so.0
/opt/openocd-0.9.0/bin/openocd
sudo cp contrib/99-openocd.rules /etc/udev/rules.d/99-openocd.rules
sudo udevadm control --reload-rules
/opt/openocd-0.9.0/bin/openocd -f /./opt/openocd-0.9.0/share/openocd/scripts/board/atmel_samr21_xplained_pro.c
```

2.6.4 Programming and Debugging Targets

2.6.4.1 Programming with avrdude

Note

avrdude uses libusb to access JTAG-ICE, Dragon, ISP etc. Section Using USB connected Programmers under Linux describes, what to do, that normal users can access these devices.

```
(1)  avrdude -P usb -p m1281 -c jtag2 -U fl:w:install/bin/sniffer_stk541.hex
(2)  avrdude -P usb:4711 -p m1281 -c jtag2 -U fl:w:install/bin/sniffer_stk541.hex
(3)  avrdude -P usb:1234 -p m1281 -c avrisp2 -U fl:w:install/bin/sniffer_stk541.hex
```

2.6.4.2 Debugging with avarice + avr-gdb (ddd)

```
avarice -I -P atmega1281 -2 -j usb --detach :4242
ddd --debugger avr-gdb install/bin/sniffer_stk541.elf &
```

When ddd/avr-gdb has opened, type "`target remote:4242<ENTER>`" in order to connect avarice and avr-gdb.

2.6.4.3 Using DebugWire

Enable DebugWire

- use any isp capable programmer: stk500v2, jtag2isp, dragon_isp??

```
ISPDUDE="-c jtag2isp -P usb -p t44"
DWDUDE="-c jtag2dw -P usb -p t44"
avrdude $ISPDUDE  -F -U hfuse:w:????:m
avrdude $DWDUDE -F
```

http://www.homebuilthardware.com/index.php/avr/linux-avrdragon-tutorial-1/

2.6.5 How to add a new radio board to the library

A detailed description on how to add new custom hardware configuration can be found at: http://uracoli.nongnu.org/custom.html .

2.6.6 USB drivers

2.6.6.1 Connecting Board and PC

Most of the current transceiver boards are equipped with a USB connector. If the USB cord is plugged into the PC, the OS will load a driver and assign a virtual serial port. Before starting the sniffer, the name of the serial port needs to be determined.

Linux:

The device names are usually:

- `/dev/ttyS[0-9]` for regular RS232 serial ports,
- `/dev/ttyUSB[0-9]` for FTDI and CP210x based boards and
- `/dev/ttyACM[0-9]` for CDC devices like the Raven RZUSB stick.

The assigned name of a serial port can be found with the command `dmesg`.

```
$ dmesg | grep tty
usb 4-1: FTDI USB Serial Device converter now attached to ttyUSB0
cdc_acm 2-3.1:1.0: ttyACM0: USB ACM device
usb 2-3.3: cp2101 converter now attached to ttyUSB1
```

Windows

The name of the assigned COM port can be found with the **Device Manager**. A quick way to open this tool (see http://en.wikipedia.org/wiki/Device_Manager) is either to type `devmgmt.msc` in `cmd.exe` window or click: `Start -> Run -> devmgmt.msc -> OK`

- FT245BM Driver
- Using the CDC driver for RZUSBSTICK
- Using USB connected Programmers under Linux

2.6.6.2 FT245BM Driver

The FTDI FT245BM is a FIFO to USB converter with a parallel interface. The USB device emulates a serial port, which is COMx in Windows or /dev/ttyUSBx in Linux (http://www.ftdichip.com/FTProducts.htm#FT245BM).

With the parallel interface its easy to provide a high data rate connection between PC and AVR. The memory buffer inside the FT245xx for both, TX and RX direction, relaxes the CPU load on the AVR side. The MCU can write/read whole chunks of data to the chip without lots of interrupts or peramnently polling a USART status register.

The FT245 can be branded with custom vendor and product IDs in a external EEPROM. In combination with the hardware described here, the following combinations can occur:

- original FT245BM ID
  - Vendor 0x0403, Device 0x6001
- Atmel STK541 Sniffer Kit
  - Vendor 0x03eb, Device 0x2009
  - http://www.nongnu.org/uracoli/hwlist.html#STK541
- Dresden Electronic Sensor Terminal Board
  - Vendor 0x1cf1, Device 0x0001
  - http://www.nongnu.org/uracoli/hwlist.html#STB230

The following sections describe, how the driver for different VID/PID in Linux and Windows is adapted:

- Patching the FT245 Driver under Linux
- Using the FT245 Driver under Windows

2.6.6.2.1 Patching the FT245 Driver under Linux

This steps have to be done in order to modify the Linux kernel module `ftdi_sio.ko` so, that it accepts devices with the vendor/device IDs of the Atmel STK541 and/or the Dresden Elektronik Sensor Terminal Board and assigns them a serial device like `/dev/ttyUSBxx`.

Modify the Driver Source Code

In the directory `/usr/src/linux/drivers/usb/serial/` there are the files `ftdi_sio.h` and `ftdi_sio.c`, which needs to be modified.

There are two patches available but most probably they will not work at a specific system, since in `ftdi_sio.c`, the line numbers change rapidly by adding new devices from other vendors. So the patches most probably needs to be applied manually.

- `ftdi_sio_2.6.13-15-default.patch` (SuSE 10.0, default kernel)
- `ftdi_sio_2.6.25.5-1.1-pae.patch` (SuSE 11.0, default kernel)

At first, the macros for the vendor and product ID of the boards have to be added in the file `ftdi_sio.h:329`.

```
#define ATMEL_VID               0x03eb  /* Atmel Vendor ID */
#define STK541_PID              0x2109  /* Zigbee Controller */
#define DE_VID                  0x1cf1  /* Dresden Electronic Vendor ID */
#define STB_PID                 0x0001  /* Sensor Terminal Board */
```

Next in the file `ftdi_sio.c` the new Vendor/Device ID tuples needs to be added:

```
static struct usb_device_id id_table_combined [] = {
        { USB_DEVICE(FTDI_VID, FTDI_AMC232_PID) },
        ...
        ...
        ...
        { USB_DEVICE(ATMEL_VID, STK541_PID) },
        { USB_DEVICE(DE_VID, STB_PID) },
        { },                                    /* Optional parameter entry */
        { }                                     /* Terminating entry */
};
```

Build the Modified Kernel Modul

```
root@pandora>cd /usr/src/linux
root@pandora>make drivers/usb/serial/ftdi_sio.ko
CC [M]  drivers/usb/serial/ftdi_sio.o
MODPOST
CC      drivers/usb/serial/ftdi_sio.mod.o
LD [M]  drivers/usb/serial/ftdi_sio.ko
```

Note

If an error message like "You have not yet configured your kernel! (missing kernel .config file)" occurs, under SuSE Linux you can quickly recover from this issue by doing "zcat /proc/config.gz > .config" in the directory `/usr/src/linux`.

Install the new Kernel Modul

```
root@pandora>rmmod ftdi_sio
root@pandora>cd /lib/modules/`uname -r`/kernel/drivers/usb/serial
root@pandora>cp ftdi_sio.ko ftdi_sio.ko.orig
root@pandora>cp /usr/src/linux/drivers/usb/serial/ftdi_sio.ko .
root@pandora>depmod -a
```

Load and Test the new Kernel Modul

```
root@pandora>modprobe ftdi_sio
root@pandora>lsmod | grep ftdi
ftdi_sio               27400  0
usbserial              28776  1 ftdi_sio
usbcore               112640  5 ftdi_sio,usbserial,uhci_hcd,rt73
root@pandora>dmesg | grep FTDI
ftdi_sio 1-1.2:1.0: FTDI USB Serial Device converter detected
usb 1-1.2: FTDI USB Serial Device converter now attached to ttyUSB0
drivers/usb/serial/ftdi_sio.c: v1.4.3:USB FTDI Serial Converters Driver
```

2.6.6.2.2 Using the FT245 Driver under Windows

2.6.6.2.3 Installation

In order to operate the sensor terminal board or STK541 under windows, the virtual COM port driver (VCP) from http://www.ftdichip.com/Drivers/VCP.htm has to be used. With the following steps the driver gets installed:

1. Download the driver package *"CDM 2.04.06 WHQL Certified.zip"*
2. Unpack it in a temp directory.
3. Apply the patch `ftdi_cdm_2.04.06.patch` manually or with the *patch* command (e.g., if mingw or cygwin is installed).
4. Connect the USB board to the PC and guide the hardware assistent to the patched directory and complete the installation. Since the WHQL certification breaks with the patch, you have explicitely to confirm, that you want to install *that* driver.
5. Now you should find the assigned COM port in the device manager.

2.6.6.2.4 Patch for ftdibus.inf and ftdiport.inf

2.6.6.2.5 VCP Driver Licensing Conditions

The Website of FTDI states http://www.ftdichip.com/Drivers/VCP.htm (2009-02-03).

FTDI device drivers may be used only in conjunction with products based on FTDI parts.
The driver may be distributed in any form as long as our license information is not modified.
If a custom Vendor ID and/or Product ID, or description string are used, it is the responsibility of the product manufacturer to maintain any changes and subsequent WHQL re-certification as a result of using these changes.

2.6.6.3 Using the CDC driver for RZUSBSTICK

The HIF driver for the Raven USB stick implements a CDC device. This can be handled by generic virtuell COM port driver.

The Vendor/Device ID used by µracoli is:

- Vendor ID: 5824 0x16C0
- Product ID: 2183 0x887

see also http://www.frank-buss.de/pid.txt

Linux

- /usr/sbin/lsusb
  - `Bus 003 Device 009:  ID 16c0:0887`
- for SuSE 10.0 the driver cdc_acm was not loaded automatically, in this case the command "modprobe cdc_acm" loads the driver.
- dmesg | grep ACM ==> "cdc_acm 3-1.4:1.0: ttyACM0: USB ACM device"
- Some older pyserial versions come up with the following error: `[Errno 11] Resource temporarily unavailable` In this case apply the patch or upgrade pyserial: http://mail.python.org/pipermail/python-list/2004-November/294242.html

Windows

- µracoli provids a INF file that installs the usbserial.sys driver. uracoli-cdc.inf

2.6.6.4 Using USB connected Programmers under Linux

SuSE 11.0

```
cd  /etc/udev/rules.d/80-usbprog.rules
vi 80-usbprog.rules
```

In this file add the following lines:

```
# AVR ISP
ATTR{idVendor}=="03eb", ATTR{idProduct}=="2104", GROUP="users", MODE="0660"
# AVR Dragon
ATTR{idVendor}=="03eb", ATTR{idProduct}=="2107", GROUP="users", MODE="0660"
# JTAG ICE MkII
ATTR{idVendor}=="03eb", ATTR{idProduct}=="2103", GROUP="users", MODE="0660"
```

If you want to enable other USB devices for users access, just check the output of lsusb command.

```
$$ lsusb
 ...
 Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
 Bus 002 Device 009: ID 03eb:2103 Atmel Corp. JTAG ICE mkII
 Bus 002 Device 003: ID 07d1:3c03 D-Link System DWL-G122 802.11g Adapter [ralink rt73]
 Bus 002 Device 002: ID 0409:005a NEC Corp. HighSpeed Hub
 ...
```

Thanks to the german description at: http://www.wiki.elektronik-projekt.de/mikrocontroller/avr/avrduc

SuSE 10.0

Add the following line at the end of file /etc/udev/rules.d/50-udev.rules

```
SUBSYSTEM=="usb", ACTION=="add", ENV{PRODUCT}=="3eb/2103/*", RUN+="/bin/sh -c '/bin/chgrp uucp $env{DEVICE}
       ; /bin/chmod g+w $env{DEVICE}'"
```

2.6.7 Random Notes

2.6.7.1 Color Codes used for Images

Group	Open Office Codes	HTML
uracoli logo	230 230 255 (Blue Gray)	#e6e6ff
radio	230 230 255 (Blue Gray)	#e6e6ff
background	247 250 247 (uracoli green)	#f7faf7
transceiver	204 204 255 (Sun 4)	#ccccff
board	230 230 230 (Gray 10%)	#e6e6e6
timer	255 204 153 (Orange 4)	#ffcc99
hif	255 128 128 (Salmon)	#ff8080
ioutil	230 230 76 (Yellow 4)	#e6e64c
sensors	174 207 0 (Diagramm 8)	#aecf00
utils	255 211 32 (Diagramm 3)	#ffd320
# free colors		
	153 204 255 (Blue 8)	#99ccff
	255 255 204 (Pale Yellow)	#ffffcc

2.6.7.2 Current Include File Layout

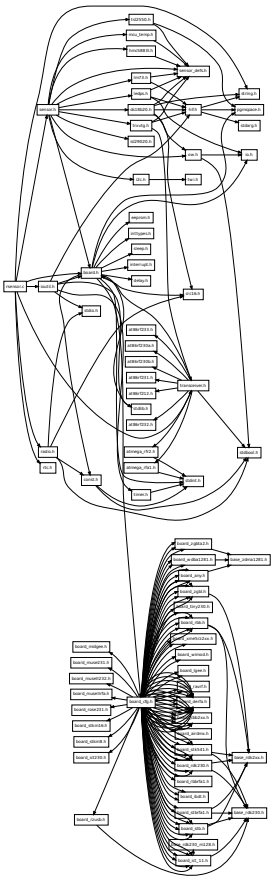


Figure 6: Include file graph for liburacoli_<board>.a

- Script Fragments
- Automatic Code Generation
- File and Directory Structure of the Package

2.6.7.3 Script Fragments

Here is a collection of tool snippets that are used during development. It took some time to find the information, so it is worth keeping this work.

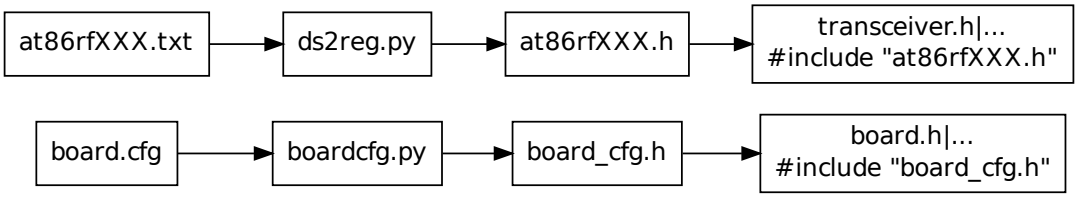
Convert Impress slides to single PNG Files

```
import os
file = "uracoli_modules.odg"
names = [None, None, "trx", "radio", "sensors", "hif", "timer", "utils", "gpio", "library", "examples", "
      applications"]
for i in range(1,len(names)+1):

    if names[i]:
        ofile = "uracoli_modules_%s" % names[i]
    else:
        ofile = "uracoli_modules"
    print "convert %s" % ofile
    cmd = 'unoconv -f pdf --stdout -e "PageRange=%d" %s > %s.pdf' % (i, file, ofile)
    print cmd
    os.system(cmd)
    os.system('convert %s.pdf -fuzz 1%% -trim +repage %s.png' % (ofile, ofile))
```

2.6.7.4 Automatic Code Generation

Some of the source files are created by code generation scripts. The following picture shows the dependencies between the templates and the generated source files.



2.6.7.5 File and Directory Structure of the Package

After unpacking the archive file, you'll find the following directory structure.

```
uracoli
|
+---Src ............................. Sources
|   +---Lib ......................... Library source code
|   |   +---Inc
|   |   |   ---boards
|   |   +---Ioutil
|   |   ---Rf230
|   +---App ......................... Application source code
|   |   +---Inc
|   |   ---Sniffer
|   ---Xmpl ......................... Example source code
|
+---Contrib ........................ PC Applications, that interact
|                                    with firmware applications
|   ---PacketCapture ............... Sniffer Interface
|
+---Doc ............................. Doxygen Sources
|   +---App
|   +---Contrib
|   +---Dev
|   +---Images
|   ---Usr
|
+---Templates
|
+---Tools
|
|  ------------------------------------------------------------------------
|     All directories below this line can be erased, they just contain
|     stuff, which is generated during the build process.
|  ------------------------------------------------------------------------
|
+---build ........................... temporary build directories
|
---install ......................... compiled/generated files
    +---bin ........................ elf/hex files
    +---doc ........................ html doc
    |   +---app
    |   +---contrib
```

```
|   +---dev
|   ---usr
+---inc ....................... include files, interface for the libs
|   ---boards
+---lib ....................... static libraries
---xmpl ....................... all example, make- and project files
```

2.7 Epilogue

- Release Notes
  An overview about the package versions.
- License
  All software and documentation provided here, is released under the terms of the Modified BSD license.
- External Package Licenses
  Licenses of txternal software packages, µracoli depends on.
- Acknowledgements
  for various contributions are stated here.

2.7.1 Release Notes

2.7.2 License and Copyright

2.7.3 External Package Licenses

`Python` is available under the Python Software Foundation License (PSF). http://docs.python.org/license.html

`pyserial` is available under the Python License. http://pyserial.sourceforge.net/appendix.html#license

`win32com` A license page could not be found on the Web.

`Wireshark` is available under the GNU General Public License version 2. http://www.wireshark.org/about.html

`µracoli` (firmware and converter application) is licensed with a Modified BSD License. http://uracoli.nongnu.org/license.html

2.7.4 Acknowledgements

- Jörg Wunsch - http://sax.sax.de/~joerg
  - for lots of tips and tricks around AVR programming
  - the one shot timer implementation, which inspired the timer function in libio_<board>.a
- Atmel Germany - http://www.atmel.com
  - for their open source 802.15.4-MAC implementation, which inspired the trx functions in libradio_<board>.a
- Peter Fleury - http://jump.to/fleury

  - for the very helpfull AVR examples and the UART Library, which where the base of the hif_uart.↩ c implementation

- Jörg Träger - http://www.icboard.de
  - for keeping up my enthusiasm and providing samples of the IN-CIRCUIT radio modules for testing.
- Andreas Bonin, Roland Blüthgen, Daniel Thiele,
  - for feedback and debugging help.

3 Module Documentation

3.1 Board Definitions

Data Structures

- struct node_config_t

3.1.1 Detailed Description

A detailed list of the supported boards can be found in section ref pgBoards

The boards are defined in the file `Config/board.cfg`

3.1.2 Functions

3.1.2.1 static uint8_t get_node_config (node_config_t * *ncfg*) [static]

Read the node_config_t structure from the end of the flash memory.

Parameters

ncfg	Pointer to the node config structure that is read from FLASH END.

Returns

Returns 0 if the crc over the structure is correct.

Examples:

xmpl_radio_range.c.

Definition at line 258 of file board.h.

3.1.2.2 static uint8_t get_node_config_eeprom (node_config_t * *ncfg*, uint8_t * *offset*) [static]

Read the node_config_t structure from an offset in the EEPROM.

Parameters

ncfg	Pointer to the node config structure that is read from EEPROM.
offset	EEPROM offset, at where the structure is read from.

Returns

Returns 0 if the crc over the structure is correct.

Definition at line 296 of file board.h.

3.1.2.3 static void jump_to_bootloader (void) [static]

Jump to the bootloader code.

Definition at line 342 of file board.h.

3.1.2.4 static void mcu_init (void) [static]

Do initialization of MCU. Here is the place for clock init, oscillator calibration, ...

Examples:

xmpl_dbg.c, xmpl_hif.c, xmpl_hif_echo.c, xmpl_i2c.c, xmpl_isl29020.c, xmpl_key_events.c, xmpl_keys.c, xmpl_leds.c, xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_lm73.c, xmpl_ow.c, xmpl_radio_range.c, xmpl_radio_stream.c, xmpl_rtc.c, xmpl_sensor.c, xmpl_timer.c, xmpl_timer_callback.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, xmpl_trx_txaret.c, and xmpl_tsl2550.c.

Definition at line 396 of file board.h.

3.1.2.5 static void store_node_config_eeprom (node_config_t * *ncfg*, uint8_t * *offset*) [static]

Read the node_config_t structure from an offset in the EEPROM.

Parameters

ncfg	Pointer to the node config structure that is written to EEPROM. The crc byte is computed within this function.
offset	EEPROM offset, at which the structure is stored.

Definition at line 325 of file board.h.

3.1.3 Data Structure Documentation

3.1.3.1 struct node_config_t

Structure for preconfigured constants that are stored at FLASHEND in program memory or in the EEPROM.

See also get_node_config(), get_node_config_eeprom() and store_node_config_eeprom().

Examples:

xmpl_radio_range.c.

Definition at line 233 of file board.h.

Data Fields

uint8_t	_reserved_[2]	For future extensions, but can be used to store user data.
uint8_t	channel	The radio channel.
uint8_t	crc	Ibutton CRC to validate if the structure is correct.
uint64_t	ieee_addr	The MAC address of the node (EUI64).
uint16_t	pan_id	The PAN ID (network ID) of the node.
uint16_t	short_addr	The short address of the node.

3.1.4 Defines

3.1.4.1 #define DEFAULT_SPI_RATE (SPI_RATE_1_2)

Typical for AVR running on 8 MHz If AVR runs on >8 MHz or synchronous to transceiver, overwrite this macro in appropiate board*.h

Also for RFA interface devices, which for them serves as dummy

Definition at line 147 of file board.h.

3.1.4.2 #define DELAY_US(x) _delay_ms(x/1000.0)

The avr-libc internal _delay_us() function allows delays up to 255us. Since the radio needs some delays, which are above this value, _delay_ms is used. As long as the argument for the DELAY_US macro is a compile time constant, no large overhead is produced, because the compiler performs the division.

Examples:

xmpl_lm73.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, xmpl_trx_txaret.c, and xmpl_tsl2550.c.

Definition at line 89 of file board.h.

3.1.4.3 #define HIF_IO_ENABLE( ) do{}while(0)

This macro is used to enable the interface circuit of the HIF

Definition at line 133 of file board.h.

3.1.4.4 #define MCU_IRQ_DISABLE cli

This must be used to construct atomic blocks

Examples:

xmpl_radio_stream.c.

Definition at line 75 of file board.h.

3.1.4.5 #define MCU_IRQ_ENABLE sei

This must be used to construct atomic blocks

Examples:

xmpl_hif.c, xmpl_hif_echo.c, xmpl_i2c.c, xmpl_isl29020.c, xmpl_key_events.c, xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_lm73.c, xmpl_ow.c, xmpl_radio_range.c, xmpl_radio_stream.c, xmpl_rtc.c, xmpl_sensor.c, xmpl_timer.c, xmpl_timer_callback.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, xmpl_trx_txaret.c, and xmpl_tsl2550.c.

Definition at line 68 of file board.h.

3.1.4.6 #define NO_HIF (1)

Macro is defined, if there is no HIF definition in appropriate board∗.h

Definition at line 127 of file board.h.

3.1.4.7 #define PULLUP_KEYS (0)

The internal pull-up resistors in the MCU are used, if this macro is defined to 1 in the board definition file

Definition at line 99 of file board.h.

3.1.4.8 #define SLEEP_ON_IDLE( )

Value:

```
do{\
            set_sleep_mode(SLEEP_MODE_IDLE);\
            sleep_mode();\
        }while(0);
```

Set MCU into idle mode.

Examples:

xmpl_key_events.c, and xmpl_rtc.c.

Definition at line 106 of file board.h.

3.1.4.9 #define TRX_RESET_HIGH( ) PORT_TRX_RESET |= MASK_TRX_RESET

set RESET pin to high level

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 158 of file board.h.

3.1.4.10 #define TRX_RESET_INIT( ) DDR_TRX_RESET |= MASK_TRX_RESET

RESET pin IO initialization

Definition at line 153 of file board.h.

3.1.4.11 #define TRX_RESET_LOW( ) PORT_TRX_RESET &= ~MASK_TRX_RESET

set RESET pin to low level

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 163 of file board.h.

3.1.4.12 #define TRX_SLPTR_HIGH( ) PORT_TRX_SLPTR |= MASK_TRX_SLPTR

set SLEEP_TR pin to high level

Examples:

xmpl_trx_echo.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 173 of file board.h.

3.1.4.13 #define TRX_SLPTR_INIT( ) DDR_TRX_SLPTR |= MASK_TRX_SLPTR

SLEEP_TR pin IO initialization

Definition at line 168 of file board.h.

3.1.4.14 #define TRX_SLPTR_LOW( ) PORT_TRX_SLPTR &= ~MASK_TRX_SLPTR

set SLEEP_TR pin to low level

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 178 of file board.h.

3.2 Transceiver Definitions

3.2.0.1 Overview

The following transceivers are supported by µracoli:

- at86rf230a.h
  - AT86RF230 Rev.A 2.4GHz IEEE 802.15.4-Transceiver.
  - http://www.atmel.com/images/doc5131.pdf
- at86rf230b.h
  - AT86RF230 Rev.B 2.4GHz IEEE 802.15.4-Transceiver.
  - http://www.atmel.com/images/doc5131.pdf
- at86rf231.h
  - AT86RF231 2.4GHz IEEE 802.15.4-2006-Transceiver.
  - http://www.atmel.com/images/doc8111.pdf
- at86rf232.h
  - AT86RF232 2.4GHz IEEE 802.15.4-2006-Transceiver.
  - http://www.atmel.com/images/doc8321.pdf
- at86rf233.h
  - AT86RF233 2.4GHz IEEE 802.15.4-2006-Transceiver.
  - http://www.atmel.com/images/atmel-8351-mcu_wireless-at86rf233_datasheet.pdf
- at86rf212.h
  - AT86RF212 - 700/868/900MHz IEEE 802.15.4-2006-Transceiver.
  - http://www.atmel.com/images/doc8168.pdf
- atmega_rfa1.h
  - ATmega128RFA1 2.4GHz IEEE 802.15.4-2006-Transceiver.
  - http://www.atmel.com/Images/Atmel-8266-MCU_Wireless-ATmega128RFA1_Datasheet.pdf
- atmega_rfr2.h
  - ATmega256RFR2 2.4GHz IEEE 802.15.4-2006-Transceiver.
  - http://www.atmel.com/images/atmel-8393-mcu_wireless-atmega256rfr2-atmega128rfr2-_datasheet.pdf

3.3 Transceiver API

The low level interface to the radio transceiver.

Data Structures

- struct trx_param_t

3.3.1 Detailed Description

3.3.1.1 Overview

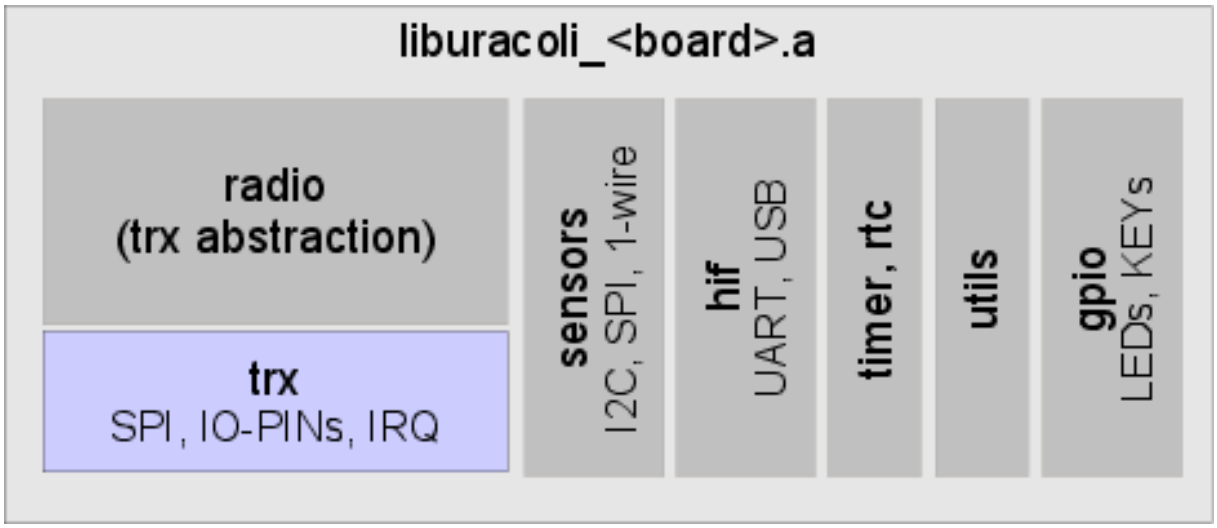


Figure 7: Tranceiver API in liburacoli_<board>.a

Usage

This group of functions provides the register level access to the radio transceivers. The `trx` interface is given in transceiver.h

The use of the Transceiver API is illustrated by the following examples.

```
#include "board.h"
#include "transceiver.h"

mcu_init();
trx_io_init(SPI_RATE_1_2);

TRX_RESET_LOW();
TRX_SLPTR_LOW();
DELAY_US(TRX_RESET_TIME_US);
TRX_RESET_HIGH();
trx_reg_write(RG_TRX_STATE,CMD_TRX_OFF);
```

Examples

- xmpl_trx_base.c
- xmpl_trx_tx.c
- xmpl_trx_rx.c

- xmpl_trx_echo.c
- xmpl_trx_txaret.c
- xmpl_trx_rxaack.c

3.3.2 Functions

3.3.2.1 trx_regval_t trx_bit_read (trx_regaddr_t *addr*, trx_regval_t *mask*, uint8_t *pos*)

Parameters

addr	offset of the register
mask	bit mask of the subregister
pos	bit position of the subregister

Return values

data	pointer where the read and demuxed value is stored

```
00001 pos = 4, mask = 0xf0
00002 register value = 0xA5
00003 *data = 0x0A
```

Examples:

xmpl_radio_range.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

3.3.2.2 void trx_bit_write (trx_regaddr_t *addr*, trx_regval_t *mask*, uint8_t *pos*, trx_regval_t *value*)

Parameters

addr	offset of the register
mask	bit mask of the subregister
pos	bit position of the subregister

Return values

value	data, which is muxed into the register

```
00001 pos = 4, mask = 0xf0
00002 register value = 0xA5 (before operation)
00003 value = 0x05
00004 register value = 0x55 (after operation)
```

Examples:

xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

3.3.2.3 uint8_t trx_decode_datarate (uint8_t *rhash*, char * *rstr*, uint8_t *nlen*)

Parameters

rhash	Hash value of a data rate.

Return values

rstr	pointer to the buffer, where data rate is copied,

Parameters

nlen	maximum length of rstr

Returns

255 in case the hashvalue `rhash` is not supported by the transceiver or invalid, otherwise 0.

3.3.2.4 void* trx_decode_datarate_p (uint8_t *rhash*)

Parameters

rhash	Hash value of a data rate.

Returns

a string pointer in the programm memory

3.3.2.5 uint8_t trx_frame_get_length (void)

Returns

length of the received frame

3.3.2.6 uint8_t trx_frame_read (uint8_t * *data,* uint8_t *datasz,* uint8_t * *lqi*)

This function reads a frame from the transceiver.

Return values

data	Pointer to an array of (Payload-) bytes that should be sent

Parameters

datasz	maximum number of bytes, which fit in the data buffer.

Return values

lqi	Pointer where the LQI value is stored

Returns

length of the downloaded frame (including the LQI byte [RADIO_TYPE == AT86RF230])

Examples:

xmpl_trx_echo.c, and xmpl_trx_rx.c.

3.3.2.7 uint8_t trx_frame_read_crc (uint8_t * *data,* uint8_t *datasz,* bool * *crc_ok*)

This function reads a frame from the transceiver. While the upload is in progress, the CRC16 value is caluculated and compared against the last two bytes. The two crc bytes **are** copied in the data buffer.

Return values

data	Pointer to an array of (Payload-) bytes that should be sent

Parameters

datasz	maximum number of bytes, which fit in the data buffer.

Return values

crc_ok	Result of the CRC16 check.

Returns

length of the downloaded frame (including the LQI byte [RADIO_TYPE == AT86RF230])

3.3.2.8 uint8_t trx_frame_read_data_crc (uint8_t * *data*, uint8_t *datasz*, uint8_t * *lqi*, bool * *crc_ok*)

This function reads a frame from the transceiver. While the upload is in progress, the CRC16 value is caluculated and compared against the last two bytes. The two crc bytes are **not** copied in the data buffer.

Return values

data	Pointer to an array of (Payload-) bytes that should be sent

Parameters

datasz	maximum number of bytes, which fit in the data buffer.

Return values

lqi	Pointer where the LQI value is stored
crc_ok	Result of the CRC16 check.

Returns

length of the downloaded frame (including the LQI byte [RADIO_TYPE == AT86RF230])

3.3.2.9 void trx_frame_write (uint8_t *length*, uint8_t * *data*)

This function writes a frame to the transceiver.

Parameters

length	Length of the frame that should be written into the frame buffer
data	Pointer to an array of (Payload-) bytes that should be sent

Note

SLP_TR! (RADIO_TYPE == AT86RF230)

Examples:

xmpl_trx_echo.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

3.3.2.10 uint8_t trx_get_datarate (void)

Returns

hash code of currently set data rate.

3.3.2.11 uint8_t trx_get_datarate_str (uint8_t *idx*, char * *rstr*, uint8_t *nlen*)

Parameters

idx	Index of the data rate.

Return values

rstr	pointer to the buffer, where data rate is copied,

Parameters

nlen	maximum length of rstr

Returns

255 in case the index `idx` is out of range, otherwise 0.

3.3.2.12 void* trx_get_datarate_str_p (uint8_t *idx*)

This function can be used to get a list of data rates, supported of the current radio transceiver.

Parameters

idx	Index of the data rate.

Returns

Program memory string pointer.

3.3.2.13 static int trx_identify (void) [static]

Returns

status value, with the following meaning:

- 0 if part and revision number match
- 1 if revision number does **not** match
- 2 if part number does **not** match
- 3 if part and revision number does **not** match

Definition at line 532 of file transceiver.h.

3.3.2.14 void trx_io_init (uint8_t *spirate*)

Parameters

spirate	Configuration Byte of the SPI Control Register (SPCR) in case of RFA1, RFR2, ... this parameter is a dummy.

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

3.3.2.15 void trx_parms_get (trx_param_t * *p*)

This function reads the static transceiver parameters, defined in a structure and protects it with a CRC16.

Parameters

p	pointer to the data structure.

3.3.2.16 uint8_t trx_parms_set (trx_param_t * p)

This function writes the static transceiver parameters, defined in a structure protects it with a CRC16.

Parameters

p	pointer to the data structure.
no_crc_check	if this parameter is true, the the CRC given in the sructure is not checked.

Returns

0 if OK, 1 if error.

3.3.2.17 uint8_t trx_reg_read (trx_regaddr_t *addr*)

This function reads a transceiver register.

Parameters

addr	Address of the Register in the Transceiver (Offset) that should be read

Returns

Contents of the Register

Examples:

xmpl_radio_range.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

3.3.2.18 void trx_reg_write (trx_regaddr_t *addr,* trx_regval_t *val*)

This function write to a transceiver register.

Parameters

addr	Address of the Register in the Transceiver (Offset) that should be written
val	Byte that will be written into the Register

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

3.3.2.19 uint8_t trx_set_datarate (uint8_t *rate_type*)

The transceiver is forced to state TRX_OFF in order to switch the data rate.

Parameters

rate_type	type code (the data rate hash codes, e.g. OQPSK250, e.g. *PSK macros) of the desired data rate.

Returns

the value of rate_type parameter or RATE_NONE

3.3.2.20 void trx_set_irq_handler (trx_irq_handler_t *irqhandler*)

set function pointer for IRQ handler

3.3.2.21 void trx_sram_read (trx_ramaddr_t *addr,* uint8_t *length,* uint8_t * *data*)

Parameters

addr	Address in the TRX's SRAM where the read burst should start
length	Length of the write burst

Return values

data	Pointer to an array of bytes that should be read

3.3.2.22 void trx_sram_write (trx_ramaddr_t *addr*, uint8_t *length*, uint8_t * *data*)

This function writes into the SRAM of the transceiver.

Parameters

addr	Address in the TRX's SRAM where the write burst should start
length	Length of the write burst
data	Pointer to an array of bytes that should be written

3.3.3 Data Structure Documentation

3.3.3.1 struct trx_param_t

Transceiver parameter structure

Definition at line 341 of file const.h.

Data Fields

unsigned int	cca: 2	CCA mode see sub register SR_CCA_MODE
channel_t	chan	current channel see sub register SR_CHANNEL
unsigned int	clkm: 3	clkm control see sub register SR_CLKM_CTRL
unsigned int	edt: 4	ED threshold see sub register SR_CCA_ED_THRES
unsigned int	txp: 4	TX power index see sub register SR_TX_PWR

3.3.4 Typedefs

3.3.4.1 typedef uint8_t ccamode_t

transceiver cca mode, 1 : ED, 2: CS, 3: CS & ED

Definition at line 334 of file const.h.

3.3.4.2 typedef int8_t channel_t

transceiver channel type

Definition at line 323 of file const.h.

3.3.4.3 typedef bool rxidle_t

radio idle state, if true radio idles in state PX_ON

Definition at line 331 of file const.h.

3.3.4.4 typedef void(* trx_irq_handler_t) (uint8_t cause)

Data Type for Transceiver IRQ callback function

Definition at line 95 of file transceiver.h.

3.3.4.5 typedef uint8_t trx_ramaddr_t

Data Type for Transceiver SRAM address

Definition at line 81 of file transceiver.h.

3.3.4.6 typedef uint8_t trx_regaddr_t

Data Type for Transceiver register address

Definition at line 89 of file transceiver.h.

3.3.4.7 typedef uint8_t trx_regval_t

Data Type for Transceiver register value

Definition at line 85 of file transceiver.h.

3.3.4.8 typedef int8_t txpwr_t

transceiver transmit type

Definition at line 326 of file const.h.

3.3.5 Defines

3.3.5.1 #define DEFAULT_PAN_ID (0xb5c2)

Default PAN-ID value used by applications

Definition at line 308 of file const.h.

3.3.5.2 #define DEFAULT_SHORT_ADDRESS (0x6172)

Default short address value used by applications

Definition at line 310 of file const.h.

3.3.5.3 #define FCTL_ACK _BV(5)

ack request in frame control field

Definition at line 296 of file const.h.

3.3.5.4 #define FCTL_DATA _BV(0)

data frame fype in frame control field

Definition at line 295 of file const.h.

3.3.5.5 #define FCTL_DST_LONG 0x0c00

destination long address in frame control field

Examples:

xmpl_radio_stream.c.

Definition at line 299 of file const.h.

3.3.5.6 #define FCTL_DST_SHORT 0x0800

destination short address in frame control field

Examples:

xmpl_radio_stream.c.

Definition at line 298 of file const.h.

3.3.5.7 #define FCTL_IPAN _BV(6)

intra pan bit in frame control field

Definition at line 297 of file const.h.

3.3.5.8 #define FCTL_SRC_LONG 0xc000

source long address in frame control field

Examples:

xmpl_radio_stream.c.

Definition at line 301 of file const.h.

3.3.5.9 #define FCTL_SRC_SHORT 0x8000

source short address in frame control field

Examples:

xmpl_radio_stream.c.

Definition at line 300 of file const.h.

3.3.5.10 #define INVALID_PART_NUM (2)

flag for invalid part number

Definition at line 113 of file transceiver.h.

3.3.5.11 #define INVALID_REV_NUM (1)

flag for invalid revision number

Definition at line 114 of file transceiver.h.

3.3.5.12 #define MAX_FRAME_SIZE (127)

Maximum size in bytes of an IEEE 802.15.4 frame

Examples:

xmpl_radio_range.c, xmpl_radio_stream.c, xmpl_trx_echo.c, and xmpl_trx_rx.c.

Definition at line 140 of file transceiver.h.

3.3.5.13 #define SPI_RATE_1_128 (3)

SPI clock running is 0.0078125 (1/128) of cpuclock

Definition at line 279 of file const.h.

3.3.5.14 #define SPI_RATE_1_16 (1)

SPI clock running is 0.0635 (1/16) of cpu clock

Definition at line 276 of file const.h.

3.3.5.15 #define SPI_RATE_1_2 (4)

SPI clock running is 0.5 (1/2) of cpu clock

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 273 of file const.h.

3.3.5.16 #define SPI_RATE_1_32 (6)

SPI clock running is 0.03125 (1/32)of cpu clock

Definition at line 277 of file const.h.

3.3.5.17 #define SPI_RATE_1_4 (0)

SPI clock running is 0.25 (1/4) of cpu clock

Definition at line 274 of file const.h.

3.3.5.18 #define SPI_RATE_1_64 (2)

SPI clock running is 0.015625 (1/64)of cpu clock

Definition at line 278 of file const.h.

3.3.5.19 #define SPI_RATE_1_8 (5)

SPI clock running is 0.125 (1/8) of cpu clock

Definition at line 275 of file const.h.

3.3.5.20 #define TRX_INIT_FAIL (1)

trx init function failed (TRX_OFF not reached after reset)

Definition at line 109 of file transceiver.h.

3.3.5.21 #define TRX_OK (0)

trx function succeeded

Definition at line 107 of file transceiver.h.

3.3.5.22 #define TRX_PLL_FAIL (2)

trx pll check function failed (PLL_LOCK coult not be observed in PLL_ON)

Definition at line 111 of file transceiver.h.

3.4 Radio API

The high level interface to the radio chip.

Data Structures

- union radio_param_t

  Container for handover of radio parameter values. More...
- struct radio_status_t

  Structure for storage of radio parameters. More...

3.4.1 Detailed Description

3.4.1.1 Overview

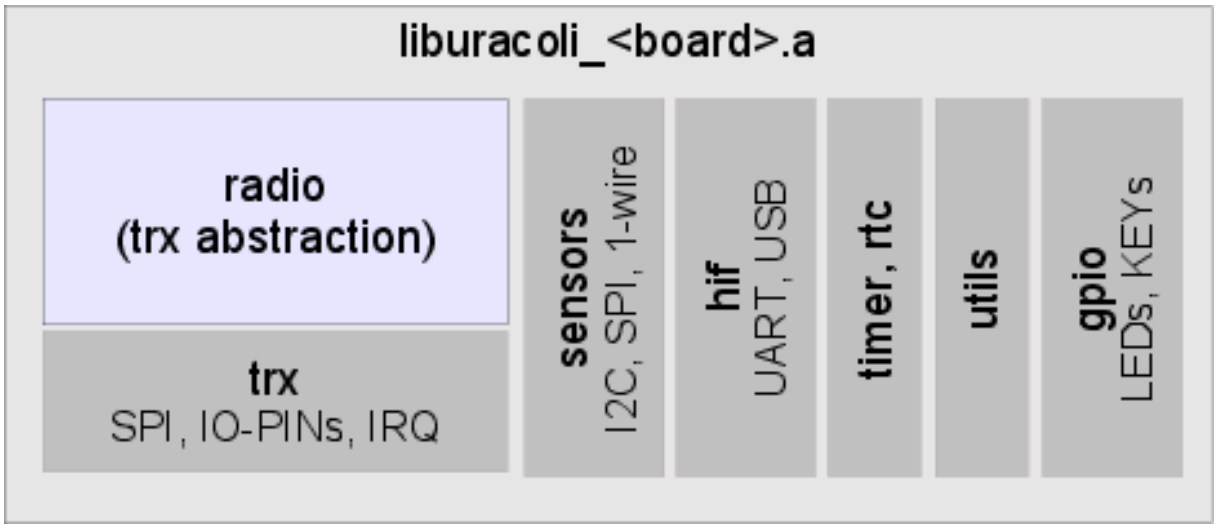


Figure 8: Radio API in liburacoli_<board>.a

Usage

```
#include "board.h"
#include "radio.h"
void main()
{
    radio_init(rxbuf, MAX_FRAME_SIZE);
    ...
}
```

Examples

- xmpl_linbuf_rx.c
- xmpl_linbuf_tx.c
- xmpl_radio_range.c
- xmpl_radio_stream.c

3.4.2 Functions

3.4.2.1 radio_cca_t radio_do_cca (void)

Returns

cca status (see radio_cca_t)

3.4.2.2 void radio_force_state (radio_state_t *state*)

Parameters

state	requested radio state

3.4.2.3 void radio_init (uint8_t * *rxbuf,* uint8_t *rxbufsz*)

The function initializes all IO ressources, needed for the usage of the radio and performs a reset.

Parameters

rxbuf	A buffer for the receive frames. This buffer needs to be static, and of size MAX_FRAME_S↩ IZE (see also function usr_radio_receive_frame).
rxbufsz	maximum size of the rx buffer, frames longer then rxbufsz will be ignored

Examples:

xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_radio_range.c, and xmpl_radio_stream.c.

3.4.2.4 void radio_send_frame (uint8_t *len,* uint8_t * *frm,* uint8_t *compcrc*)

Initiates a frame transmission procedure, and then downloads the frame passed as `frm`. The function returns immediately once the frame download procedure completed; the end of transmission is signalled by the usr_radio↩ _tx_done callback. If the radio is in STATE_TXAUTO, a full unslotted CSMA-CA is procedure is performed by the transceiver hardware, including frame retransmissions in case the transmitted frame has requested an acknowledgement by the recipient. If the transceiver is in STATE_TX, an immediate frame transmission is initiated, without CSMA-CA or frame retransmissions.

Parameters

len	length of frame to transmit
frm	pointer to frame to transmit
compcrc	compute CRC-16 if != 0 (currently ignored)

Examples:

xmpl_linbuf_tx.c, xmpl_radio_range.c, and xmpl_radio_stream.c.

3.4.2.5 void radio_set_param (radio_attribute_t *attr,* radio_param_t *parm*)

Note that the radio must not be in STATE_SLEEP when setting parameters. Preferrably, it should be kept in STA↩ TE_OFF (in which it is immediately after a call to radio_init) in order to set parameters.

Parameters

attr	attribute parameter (enumeration value radio_attribute_t)
parm	pointer to parameter value (union type radio_param_t)

Examples:

```
00001 radio_set_param(RP_IDLESTATE(STATE_RX));
```

```
00001 #define CHANNEL (17)
00002
00003 radio_set_param(RP_CHANNEL(CHANNEL));
```

Examples:

xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_radio_range.c, and xmpl_radio_stream.c.

3.4.2.6 void radio_set_state (radio_state_t *state*)

Parameters

state	requested radio state

Examples:

xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_radio_range.c, and xmpl_radio_stream.c.

3.4.2.7 void usr_radio_error (radio_error_t *err*)

Error callback function which has to be implemented in the application.

This function is called, when a fatal error occurs. see also radio_error_t.

Parameters

err	error code being reported

3.4.2.8 void usr_radio_irq (uint8_t *cause*)

Interrupt callback function.

Parameters

cause	value of the transceiver's IRQ status register

3.4.2.9 uint8_t* usr_radio_receive_frame (uint8_t *len,* uint8_t * *frm,* uint8_t *lqi,* int8_t *ed,* uint8_t *crc_fail*)

This function is called within an interrupt context for every received frame, according to the current receive mode used (which involves address filtering in case the transceiver is in state RX_AUTO). The frame has been internally stored into the receive buffer that has been configured before, either by radio_init, or as the result of a previous call to usr_radio_receive_frame. The function must return a valid pointer to a receive buffer to be used for receiving the next frame; this can be the same value as the parameter `frm`, or a different one in case the application wants to maintain multiple buffers.

The `rssi` parameter is obtained at the frame's RX_START interrupt. If there was no RX_START interrupt to read an RSSI value at, the value VOID_RSSI is passed instead.

Parameters

len	length of frame received
frm	pointer to frame received
lqi	LQI value reported by transceiver
rssi	RSSI value obtained from transceiver
crc_fail	boolean indicating whether the received frame failed FCS verification

Returns

address of new receive buffer

Examples:

xmpl_linbuf_rx.c, xmpl_radio_range.c, and xmpl_radio_stream.c.

3.4.2.10 void usr_radio_tx_done (radio_tx_done_t *status*)

Transmit done callback function.

Parameters

status	completion status, radio_tx_done_t

Examples:

xmpl_radio_range.c.

3.4.3 Data Structure Documentation

3.4.3.1 union radio_param_t

Note

The elements in this union should have max. a size of 2 byte, all other parameters should be configured with the void pointer.

Definition at line 165 of file radio.h.

Data Fields

ccamode_t	cca_mode	Value for cca mode.
channel_t	channel	Value for current radio channel. (MIN_CHANNEL ... MAX_CHANNEL)
uint8_t	data_rate	data rate type
radio_state_t	idle_state	after TX go to idle state
uint64_t *	long_addr	Pointer to long (64-bit) address
uint16_t	pan_id	Value for PANID
uint8_t	rx_lna	RX LNA type
uint16_t	short_addr	Value for short address
uint8_t	tx_pa	TX power amp type
txpwr_t	tx_pwr	Value for transmit power in dB.

3.4.3.2 struct radio_status_t

Definition at line 219 of file radio.h.

Data Fields

uint8_t	cca_mode	Current cca mode.
uint8_t	channel	Current radio channel. (MIN_CHANNEL ... MAX_CHANNEL)
radio_state_t	idle_state	after TX go to idle state
uint8_t	rx_lna	
uint8_t *	rxframe	Pointer for frame data storage.
uint8_t	rxframesz	Length of the buffer rxframesz
radio_state_t	state	Current transceiver state.
uint8_t	tx_pa	
uint8_t	tx_pwr	Current transmit power.

3.4.4 Typedefs

3.4.4.1 typedef uint8_t radio_state_t

Radio state type, supported values are

- STATE_OFF,
- STATE_TX,
- STATE_TXAUTO,

- STATE_RX,
- STATE_RXAUTO,
- STATE_SLEEP.

Definition at line 78 of file radio.h.

3.4.5 Defines

3.4.5.1 #define MOD_BPSK_20 (0)

PHY modulation BPSK, 20 kbit/s

Definition at line 197 of file const.h.

3.4.5.2 #define MOD_BPSK_40 (1)

PHY modulation BPSK, 40 kbit/s

Definition at line 198 of file const.h.

3.4.5.3 #define MOD_OQPSK_100 (2)

PHY modulation O-QPSK, 100 kbit/s

Definition at line 199 of file const.h.

3.4.5.4 #define MOD_OQPSK_1000 (7)

PHY modulation O-QPSK, 1000 kbit/s

Definition at line 204 of file const.h.

3.4.5.5 #define MOD_OQPSK_200 (3)

PHY modulation O-QPSK, 200 kbit/s

Definition at line 200 of file const.h.

3.4.5.6 #define MOD_OQPSK_2000 (8)

PHY modulation O-QPSK, 2000 kbit/s

Definition at line 205 of file const.h.

3.4.5.7 #define MOD_OQPSK_250 (4)

PHY modulation O-QPSK, 250 kbit/s

Definition at line 201 of file const.h.

3.4.5.8 #define MOD_OQPSK_400 (5)

PHY modulation O-QPSK, 400 kbit/s

Definition at line 202 of file const.h.

3.4.5.9 #define MOD_OQPSK_500 (6)

PHY modulation O-QPSK, 500 kbit/s

Definition at line 203 of file const.h.

3.4.5.10 #define RADIO_AT86RF212 (4)

Identifier for radio AT86RF212

Definition at line 181 of file const.h.

3.4.5.11 #define RADIO_AT86RF230 (1)

Identifier for radio AT86RF230

Definition at line 177 of file const.h.

3.4.5.12 #define RADIO_AT86RF230A (RADIO_AT86RF230)

Identifier for radio AT86RF230 Rev A

Definition at line 178 of file const.h.

3.4.5.13 #define RADIO_AT86RF230B (2)

Identifier for radio AT86RF230 Rev B

Definition at line 179 of file const.h.

3.4.5.14 #define RADIO_AT86RF231 (3)

Identifier for radio AT86RF231

Definition at line 180 of file const.h.

3.4.5.15 #define RADIO_AT86RF232 (9)

Identifier for radio AT86RF232

Definition at line 186 of file const.h.

3.4.5.16 #define RADIO_AT86RF233 (10)

Identifier for radio AT86RF233

Definition at line 187 of file const.h.

3.4.5.17 #define RADIO_ATMEGA128RFA1_A (5)

Identifier for radio ATmega128RFA1 Rev. A

Definition at line 182 of file const.h.

3.4.5.18 #define RADIO_ATMEGA128RFA1_B (6)

Identifier for radio ATmega128RFA1 Rev. B

Definition at line 183 of file const.h.

3.4.5.19 #define RADIO_ATMEGA128RFA1_C (7)

Identifier for radio ATmega128RFA1 Rev. C

Definition at line 184 of file const.h.

3.4.5.20 #define RADIO_ATMEGA128RFA1_D (8)

Identifier for radio ATmega128RFA1 Rev. D

Definition at line 185 of file const.h.

3.4.5.21 #define RADIO_ATMEGA2564RFR2 (12)

Identifier for radio ATmega128RFR2

Definition at line 189 of file const.h.

3.4.5.22 #define RADIO_ATMEGA256RFR2 (11)

Identifier for radio ATmega128RFR2

Definition at line 188 of file const.h.

3.4.5.23 #define RADIO_BAND_2400 (4)

2.4GHz frequency band (international)

Definition at line 194 of file const.h.

3.4.5.24 #define RADIO_BAND_700 (1)

700MHz frequency band (china)

Definition at line 191 of file const.h.

3.4.5.25 #define RADIO_BAND_800 (2)

868MHz frequency band (europe)

Definition at line 192 of file const.h.

3.4.5.26 #define RADIO_BAND_900 (3)

900MHz frequency band (north america)

Definition at line 193 of file const.h.

3.4.5.27 #define RADIO_CFG_DATA {chan: 16, txp: 0, cca: 1, edt: 11, clkm: 0, crc: 0xab12}

a default radio configuration data structure

Definition at line 370 of file radio.h.

3.4.5.28 #define RADIO_CFG_EEOFFSET (8)

offset of radio config data in EEPROM

Definition at line 365 of file radio.h.

3.4.5.29 #define RP_CCAMODE(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the CCA mode to x.

Definition at line 281 of file radio.h.

3.4.5.30 #define RP_CHANNEL(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the channel number to x.

Helper macro to construct the arguments for radio_set_param in order to set the TX power amplifier enable number to x.

Examples:

xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_radio_range.c, and xmpl_radio_stream.c.

Definition at line 342 of file radio.h.

3.4.5.31 #define RP_CHANNEL(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the channel number to x.

Helper macro to construct the arguments for radio_set_param in order to set the TX power amplifier enable number to x.

Definition at line 342 of file radio.h.

3.4.5.32 #define RP_DATARATE(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the data rate to x.

Examples:

xmpl_radio_range.c.

Definition at line 330 of file radio.h.

3.4.5.33 #define RP_IDLESTATE(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the transceiver's idle state to x.

Examples:

xmpl_radio_range.c, and xmpl_radio_stream.c.

Definition at line 269 of file radio.h.

3.4.5.34 #define RP_LONGADDR(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the long address pointer to x.

Definition at line 317 of file radio.h.

3.4.5.35 #define RP_PANID(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the PAN ID to x.

Examples:

xmpl_radio_range.c.

Definition at line 293 of file radio.h.

3.4.5.36 #define RP_RX_LNA(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the TX power amplifier enable number to x.

Definition at line 354 of file radio.h.

3.4.5.37 #define RP_SHORTADDR(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the short address to x.

Examples:

xmpl_radio_range.c.

Definition at line 305 of file radio.h.

3.4.5.38 #define RP_TXPWR(*x*)

Helper macro to construct the arguments for radio_set_param in order to set the tx power value to x.

Definition at line 257 of file radio.h.

3.4.5.39 #define STATE_OFF (0)

Radio state enumerationTRX in OFF state.

Examples:

HelloRadio.ino, IoRadio.ino, xmpl_radio_range.c, and xmpl_radio_stream.c.

Definition at line 58 of file radio.h.

3.4.5.40 #define STATE_RX (2)

Basic mode RX state.

Examples:

xmpl_linbuf_rx.c, xmpl_radio_stream.c, and xmpl_trx_echo.c.

Definition at line 62 of file radio.h.

3.4.5.41 #define STATE_RXAUTO (4)

Extended mode RX state (RX_AACK).

Examples:

xmpl_radio_range.c.

Definition at line 66 of file radio.h.

3.4.5.42 #define STATE_SLEEP (5)

Sleep state (lowest power consumption).

Examples:

Remote.ino.

Definition at line 68 of file radio.h.

3.4.5.43 #define STATE_TX (1)

Basic mode TX state.

Examples:

xmpl_linbuf_tx.c, xmpl_radio_stream.c, and xmpl_trx_echo.c.

Definition at line 60 of file radio.h.

3.4.5.44 #define STATE_TXAUTO (3)

Extended mode TX state (TX_ARET).

Examples:

xmpl_radio_range.c.

Definition at line 64 of file radio.h.

3.4.5.45 #define VOID_RSSI (0xff)

Code for invalid RSSI value.

Definition at line 238 of file radio.h.

3.4.6 Enums

3.4.6.1 enum radio_attribute_t

Enumeration to identify radio attributes.

Enumerator

phyCurrentChannel Set the current channel

phyChannelsSupported Currently unused

phyTransmitPower Set the Tx power

phyIdleState Transceiver state to return to after transmission

phyCCAMode CCA mode to use in CSMA-CA:

value	CCA mode
0	carrier sense OR energy above threshold
1	energy above threshold (default)
2	carrier sense
3	carrier sense AND energy above threshold

phyPanId PAN ID to use in STATE_RXAUTO frame filter

phyShortAddr Short (16-bit) address to use in STATE_RXAUTO frame filter

phyLongAddr Pointer to long (EUI-64) address to use in STATE_RXAUTO frame filter

phyDataRate Datarate

phyTxPa PA enable

phyRxLna LNA enable

Definition at line 119 of file radio.h.

3.4.6.2 enum radio_cca_t

codes for CCA

Enumerator

RADIO_CCA_FREE The CCA measurement estimates, that the channel is free.

RADIO_CCA_BUSY The CCA measurement estimates, that the channel is busy.

RADIO_CCA_FAIL The CCA measurement was not finished.

Definition at line 95 of file radio.h.

3.4.6.3 enum radio_error_t

Enumerator

SUCCESS OK Code

STATE_SET_FAILED function radio_set_state failed

SET_PARM_FAILED function radio_set_param failed

GET_PARM_FAILED function radio_get_param failed

GENERAL_ERROR something unexpected happened

Definition at line 203 of file radio.h.

3.4.6.4 enum radio_tx_done_t

error codes for tx done event

Enumerator

TX_OK transmission completed successfully

TX_CCA_FAIL channel was busy (TX_AUTO only)

TX_NO_ACK no ACK received (TX_AUTO only)

TX_FAIL unexpected error

Definition at line 83 of file radio.h.

3.5 Timer/RTC API

Starting, stopping and execution of time triggered actions.

Data Structures

- struct time_stamp_t

3.5.1 Detailed Description

3.5.1.1 Overview

Usage

The timer module is inspired by Jörg Wunschs timer implementation, which can be found here: http://sax.↩ sax.de/~joerg/avr-timer/

RTC functions are implemented by use of the asynchronous clock facilities of the microcontroller.

Examples

- xmpl_timer.c
- xmpl_timer_callback.c
- xmpl_rtc.c

3.5.2 Functions

3.5.2.1 void rtc_init (void(*)(void) *tickfunc*)

RTC initialisation

Parameters

tf	pointer to rtc callback function.

Examples:

xmpl_rtc.c.

3.5.2.2 void rtc_start (void)

This function starts the RTC

Examples:

xmpl_rtc.c.

3.5.2.3 void rtc_stop (void)

This function stops the RTC

3.5.2.4 void timer_get_tstamp (time_stamp_t * *ts*)

Function that returns the internal system time counters as "libpcap" compatible time stamp.

Note

This routine takes ~548 cycles for execution. In case of a 8Mhz driven system, this is a 68.5 us.

Parameters

ts	timestamp data structure

3.5.2.5 void timer_init (void)

This function is responsible for the setup of the hardware timers.

Examples:

xmpl_radio_range.c, and xmpl_timer_callback.c.

3.5.2.6 timer_hdl_t timer_restart (timer_hdl_t *th*, time_t *duration*)

If the timer is found in the timer queue, then it is rstarted with the new duration value.

Parameters

th	Handle of the timer. The timer needs to exist in the timer queue, e.g. it is started with timer_start() and not yet expired.
duration	time in system ticks from now, when the timer expires.

Returns

the value of NONE_TIMER if the timer could not be found in the timer queue. Otherwise the value of th.

3.5.2.7 timer_hdl_t timer_start (timer_handler_t * *thfunc*, time_t *duration*, timer_arg_t *arg*)

This function initially creates a timer and assigns a timer handle to it. The timer handle is reference number, which identifies the timer uniquely and is needed for restart and stop a running timer.

Parameters

thfunc	pointer to a function, which is called when the timer expires.
duration	time in system ticks from now, when the timer expires.
arg	argument, which is passed to the timer function.

Returns

the value of NONE_TIMER if the timer could not be started. Otherwise a handle != NONE_TIMER, which is needed for restarting and stopping the timer.

Examples:

xmpl_radio_range.c, and xmpl_timer_callback.c.

3.5.2.8 timer_hdl_t timer_stop (timer_hdl_t *th*)

Parameters

th	Handle of the timer. The timer needs to exist in the timer queue, e.g. it is started with timer_start() and not yet expired.

Returns

the value of NONE_TIMER if the timer could not be found in the timer queue. Otherwise the value of th.

3.5.3 Data Structure Documentation

3.5.3.1 struct time_stamp_t

High resolution time stamp in seconds and micro seconds

Definition at line 70 of file timer.h.

3.5.4 Typedefs

3.5.4.1 typedef uint32_t time_t

Data type for time values (measured in number of system ticks).

Definition at line 64 of file timer.h.

3.5.4.2 typedef uint32_t timer_arg_t

Data type for the argument of a timer handler function.

Definition at line 80 of file timer.h.

3.5.4.3 typedef time_t(timer_handler_t) (timer_arg_t p)

Data type for timer expiration action function. This function is called, when the expiration time is over. When luanched, the function is called with a parameter p of type .timer_arg_t. If the function returns a value, which is greate then 0, the timer is restarted again.

Definition at line 95 of file timer.h.

3.5.4.4 typedef uint16_t timer_hdl_t

Data type for a timer handle (a reference number to identify a running timer).

Definition at line 86 of file timer.h.

3.5.5 Defines

3.5.5.1 #define MSEC(v) ((time_t)(v / (1.0e3 * TIMER_TICK)))

Macro that converts the millisecond value v into TIMER_IRQ_vect ticts

Examples:

xmpl_timer.c, and xmpl_timer_callback.c.

Definition at line 48 of file timer.h.

3.5.5.2 #define NONE_TIMER (0)

Symbolic name for invalid timer handle

Definition at line 98 of file timer.h.

3.5.5.3 #define USEC(v) ((time_t)(v / (1.0e6 * TIMER_TICK)))

Macro that converts the microsecond value v into TIMER_IRQ_vect ticts

Definition at line 50 of file timer.h.

3.6 HostInterface API

Communicating with the host computer.

3.6.1 Detailed Description

3.6.1.1 Overview

Usage

Examples

- xmpl_hif.c
- xmpl_hif_echo.c

3.6.2 Functions

3.6.2.1 void hif_dump (uint16_t *sz*, uint8_t * *d*)

Parameters

sz	number of bytes, that will be dumped.
d	pointer to the data array, that will be dumped.

Examples:

xmpl_i2c.c.

3.6.2.2 void hif_echo (FLASH_STRING_T *str*)

Parameters

str	string, which is located in flash memory

Examples:

xmpl_hif.c, and xmpl_hif_echo.c.

3.6.2.3 uint8_t hif_get_blk (unsigned char * *data*, uint8_t *max_size*)

Parameters

data	buffer where the bytes are stored
max_size	maximum number of bytes, which can be stored in the buffer.

Returns

number of bytes stored in the buffer

3.6.2.4 static int hif_get_dec_number (void) [static]

Returns

integer value of the negative number.

Examples:

xmpl_isl29020.c, and xmpl_lm73.c.

Definition at line 192 of file hif.h.

3.6.2.5 static uint16_t hif_get_number (int8_t *base*) [static]

Enter a integer number.

Parameters

base	base value for the number conversion, e.g. 10 for entering a decimal number.

Returns

16 bit unsigned integer value.

Definition at line 237 of file hif.h.

3.6.2.6 int hif_getc (void)

Returns

The Character or EOF in case of error or end-of-file

Examples:

xmpl_hif.c, xmpl_hif_echo.c, xmpl_i2c.c, xmpl_isl29020.c, xmpl_lm73.c, xmpl_ow.c, and xmpl_tsl2550.c.

3.6.2.7 void hif_init (const uint32_t *baudrate*)

Parameters

baudrate	data rate of the interface in bit/s

Examples:

xmpl_hif.c, xmpl_hif_echo.c, xmpl_i2c.c, xmpl_isl29020.c, xmpl_linbuf_rx.c, xmpl_lm73.c, xmpl_ow.c, xmpl_↩radio_range.c, xmpl_sensor.c, xmpl_timer_callback.c, and xmpl_tsl2550.c.

3.6.2.8 void hif_printf (FLASH_STRING_T *fmt,* ...)

Parameters

fmt	format string, which is located in flash memory
...	variable argument list

Examples:

xmpl_hif.c, and xmpl_hif_echo.c.

3.6.2.9 uint8_t hif_put_blk (unsigned char * *data,* uint8_t *size*)

Parameters

data	pointer to the data array.
size	size of the block.

Returns

num number of bytes, which was send.

Examples:

xmpl_hif.c, and xmpl_linbuf_rx.c.

3.6.2.10 int hif_putc (int *c*)

Parameters

data	Character to send

Returns

The Character or EOF in case of error

Examples:

xmpl_hif.c, xmpl_hif_echo.c, xmpl_i2c.c, and xmpl_ow.c.

3.6.2.11 void hif_puts (const char * *s*)

Parameters

s	pointer to a null terminated string, which is located in RAM.

Examples:

xmpl_i2c.c.

3.6.2.12 void hif_puts_p (const char * *progmem_s*)

Parameters

progmem_s	pointer to a null terminated string, which is located in program memory.

3.6.2.13 static int hif_split_args (char * *txtline*, int *maxargs*, char ** *argv*) [static]

This function creates argc,argv style data from a null terminated string. The splitting is done on the base of spaces (ASCII 32).

Parameters

txtline	string to split
maxargs	maximum number of arguments to split

Return values

argv	array of pointers, that store the arguments

Returns

number of arguments splitted (argc)

Examples:

xmpl_i2c.c, and xmpl_ow.c.

Definition at line 162 of file hif.h.

3.6.3 Defines

3.6.3.1 #define DUMP(*sz*, *ptr*) hif_dump(sz,ptr)

Wrapper macro for hif_dump()

Examples:

xmpl_hif.c.

Definition at line 58 of file hif.h.

3.6.3.2 #define HIF_AT90USB (31)

Identifier for HIF type USB/ATmega1287

Definition at line 240 of file const.h.

3.6.3.3 #define HIF_FT245 (30)

Identifier for HIF type USB/FT245

Definition at line 239 of file const.h.

3.6.3.4 #define HIF_NONE (0)

Identifier for no host interface

Definition at line 229 of file const.h.

3.6.3.5 #define HIF_SERCOM0 (40)

Identifier for HIF type SERCOM0

Definition at line 243 of file const.h.

3.6.3.6 #define HIF_UART_0 (10)

Identifier for HIF type UART 0

Definition at line 232 of file const.h.

3.6.3.7 #define HIF_UART_1 (11)

Identifier for HIF type UART 1

Definition at line 233 of file const.h.

3.6.3.8 #define HIF_USARTD0 (20)

Identifier for HIF type USARTD0

Definition at line 236 of file const.h.

3.6.3.9 #define HIF_USARTE0 (21)

Identifier for HIF type USARTD0

Definition at line 237 of file const.h.

3.6.3.10 #define PRINT(*fmt*) hif_echo(FLASH_STRING(fmt))

Wrapper macro for hif_echo()

Examples:

xmpl_i2c.c, xmpl_isl29020.c, xmpl_lm73.c, xmpl_ow.c, and xmpl_tsl2550.c.

Definition at line 56 of file hif.h.

3.6.3.11 #define PRINTF(*fmt, ...*) hif_printf(FLASH_STRING(fmt), __VA_ARGS__)

Wrapper macro for hif_printf()

Examples:

xmpl_hif.c, xmpl_hif_echo.c, xmpl_i2c.c, xmpl_isl29020.c, xmpl_lm73.c, xmpl_ow.c, xmpl_radio_range.c, xmpl_sensor.c, xmpl_timer_callback.c, and xmpl_tsl2550.c.

Definition at line 54 of file hif.h.

3.6.3.12 #define URACOLI_USB_PID (2183)

uracoli USB Product ID see also USB_VID_URACOLI

Definition at line 259 of file const.h.

3.6.3.13 #define URACOLI_USB_VID (5824)

uracoli USB Vendor ID

The pair of the uracoli vendor id, device id are obtained by

- http://www.voti.nl/pids/
- http://www.frank-buss.de/pid.txt

Definition at line 254 of file const.h.

3.7 GPIO API

Controlling the LEDs and Keys.

3.7.1 Detailed Description

3.7.1.1 Overview

Usage

Examples

- xmpl_key_events.c
- xmpl_keys.c
- xmpl_leds.c
- xmpl_dbg.c

3.7.2 Functions

3.7.2.1 static uint8_t keys_debounced (void) [static]

Returns

status ot the debounced key

Examples:

xmpl_keys.c, and xmpl_radio_range.c.

Definition at line 249 of file ioutil.h.

3.7.2.2 static void trap_if_key_pressed (void) [static]

Returns

status ot the debounced key

Examples:

xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 286 of file ioutil.h.

3.7.3 Defines

3.7.3.1 #define KEY_GET( ) (0)

Reading of the KEY port directly and return the value LSB aligbed.

Examples:

xmpl_key_events.c.

Definition at line 221 of file ioutil.h.

3.7.3.2 #define KEY_INIT( )

Initialisation of the KEY port

Examples:

xmpl_key_events.c, xmpl_keys.c, and xmpl_radio_range.c.

Definition at line 218 of file ioutil.h.

3.7.3.3 #define LED_CLR(*ln*) do{}while(0)

Switch the LED with the number *ln* OFF.

Examples:

xmpl_leds.c, xmpl_rtc.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 175 of file ioutil.h.

3.7.3.4 #define LED_GET_VALUE( ) 0

Read back the current numeric value from the LED port.

Examples:

xmpl_key_events.c, and xmpl_trx_rx.c.

Definition at line 151 of file ioutil.h.

3.7.3.5 #define LED_INIT( ) do{}while(0)

Initialisation of the LED port.

Examples:

xmpl_hif_echo.c, xmpl_i2c.c, xmpl_isl29020.c, xmpl_key_events.c, xmpl_keys.c, xmpl_leds.c, xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_lm73.c, xmpl_ow.c, xmpl_radio_range.c, xmpl_radio_stream.c, xmpl_rtc.c, xmpl_sensor.c, xmpl_timer.c, xmpl_timer_callback.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, xmpl_trx_txaret.c, and xmpl_tsl2550.c.

Definition at line 116 of file ioutil.h.

3.7.3.6 #define LED_MAX_VALUE ((1<<LED_NUMBER)-1)

Maximum value, that can be displayed on the LEDs

Examples:

xmpl_leds.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 205 of file ioutil.h.

3.7.3.7 #define LED_NUMBER (0)

Number of LEDs for this board

Examples:

xmpl_leds.c.

Definition at line 111 of file ioutil.h.

3.7.3.8 #define LED_SET(*ln*) do{}while(0)

Switch the LED with the number *ln* ON.

Examples:

xmpl_i2c.c, xmpl_isl29020.c, xmpl_leds.c, xmpl_lm73.c, xmpl_ow.c, xmpl_rtc.c, xmpl_sensor.c, xmpl_trx_base.c, xmpl_trx_tx.c, xmpl_trx_txaret.c, and xmpl_tsl2550.c.

Definition at line 163 of file ioutil.h.

3.7.3.9 #define LED_SET_VALUE(*x*) do{}while(0)

Display a numeric value on the LED port. The value x is masked out according LED_MASK, so that depending on the number of LEDs of the board the maximum displayed value is LED_MAX_VALUE.

Examples:

xmpl_key_events.c, xmpl_keys.c, xmpl_leds.c, xmpl_trx_base.c, xmpl_trx_echo.c, xmpl_trx_rx.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, and xmpl_trx_txaret.c.

Definition at line 135 of file ioutil.h.

3.7.3.10 #define LED_TOGGLE(*ln*) do{}while(0)

Toggle the LED with the number *n*.

Examples:

xmpl_i2c.c, xmpl_isl29020.c, xmpl_key_events.c, xmpl_leds.c, xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, xmpl_lm73.c, xmpl_radio_range.c, xmpl_radio_stream.c, xmpl_timer.c, xmpl_timer_callback.c, xmpl_trx_base.c, xmpl_trx_rxaack.c, xmpl_trx_tx.c, xmpl_trx_txaret.c, and xmpl_tsl2550.c.

Definition at line 199 of file ioutil.h.

3.7.3.11 #define LED_VAL(*msk, val*) do{}while(0)

.....

Definition at line 186 of file ioutil.h.

3.8 Utilities API

Utility functions

Data Structures

- struct buffer_t

3.8.1 Detailed Description

3.8.1.1 Overview

Usage

Examples

- xmpl_linbuf_rx.c
- xmpl_linbuf_tx.c

3.8.2 Functions

3.8.2.1 uint8_t buffer_append_block (buffer_t * *b*, void * *pdata*, uint8_t *size*)

append a data block at the end of a buffer

Examples:

xmpl_linbuf_rx.c, and xmpl_radio_stream.c.

3.8.2.2 int buffer_append_char (buffer_t * *b*, uint8_t *c*)

append a char at the end of a buffer

Examples:

xmpl_linbuf_tx.c.

3.8.2.3 uint8_t buffer_get_block (buffer_t * *b*, void * *pdata*, uint8_t *size*)

read a datablock from the start of a buffer

3.8.2.4 int buffer_get_char (buffer_t * *b*)

read a char from the start of a buffer

Examples:

xmpl_linbuf_rx.c.

3.8.2.5 buffer_t* buffer_init (void * *pmem*, uint8_t *size*, uint8_t *start*)

format a chunk of memory as buffer_t structure

Examples:

xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, and xmpl_radio_stream.c.

3.8.2.6 uint8_t buffer_prepend_block (buffer_t * b, void * pdata, uint8_t size)

prepend a data block at the start of a buffer

Examples:

xmpl_linbuf_tx.c, and xmpl_radio_stream.c.

3.8.2.7 int buffer_prepend_char (buffer_t * b, int c)

prepend a char at the start of a buffer

3.8.3 Data Structure Documentation

3.8.3.1 struct buffer_t

buffer structure, which supports appending and prepending of data as well as chaining other buffers.

Examples:

xmpl_linbuf_rx.c, xmpl_linbuf_tx.c, and xmpl_radio_stream.c.

Definition at line 72 of file ioutil.h.

Data Fields

uint8_t	data[ ]	data block
uint8_t	iend	index of the end of the data block
uint8_t	istart	index of the start of the data block
uint8_t	len	total lenght of the data block
void *	next	pointer to next buffer
uint8_t	used	tag if buffer is used or free

3.9 Sensor Drivers

Low Level Sensor-Bus and Sensor functions.

Modules

- I2C Bus Driver
- One Wire Bus Driver
- DS18B20 - One Wire Temperature Sensor.
- HMC5883L - 3-Axis Digital Compass IC
- ISL 29020 - Light Sensor
- LEDPS - Using a LED as Photo Sensor
- LM73 - Temperature Sensor
- MCU Temperature Sensor
- MCU Operating Voltage Sensor
- TSL2550 - Ambient Light Sensor

3.9.1 Detailed Description

3.9.2 Overview

Examples

- Sensor Bus Drivers
  - xmpl_ow.c
  - xmpl_i2c.c
- Sensor Drivers
  - xmpl_isl29020.c
  - xmpl_lgee_acc_simple.c
  - xmpl_tsl2550.c
  - xmpl_lm73.c

3.10 Sensor API

High Level Sensor functions.

Data Structures

- struct board_sensor_ctx_t

3.10.1 Detailed Description

3.10.1.1 Overview

The sensor API implements a generic abstraction for the sensors that are supported by µracoli. The low level drivers are documented in section Sensor Drivers.

The sensors of a board are defined in file `board.cfg` in the key `sensors:` <sensors1> <sensor2> ...

Usage

```
uint8_t nb_sensors;
nb_sensors = create_board_sensors();
/* beside the return value of create_board_sensors() the number of sensors can be retrieved with
       sensor_get_number() too */
PRINTF("number of sensors: %d\n", sensor_get_number());
for (i = 0; i < nb_sensors; i++)
{
    PRINTF("  %d : id=%d name=%s type=%s data_sz=%d\n",
           i, sensor_get_id(i),
           sensor_get_name(i),
           sensor_get_type(i),
           sensor_get(i, NULL));
}
while (1)
{
    sensor_trigger(ALL_SENSORS, 1);
    DELAY_MS(10); // wait for conversion ready
    sensor_getALL_SENSORS, buf);
    PRINTF("decode: %s\n",sensor_decode(buf, line, sizeof(line)));
    DELAY_MS(2000);
}
```

Examples

- High level abstraction example xmpl_sensor.c

3.10.2 Functions

3.10.2.1 uint8_t create_board_sensors (void)

This function creates the data structure for all sensors that are defined per board.

Examples:

xmpl_sensor.c.

3.10.2.2 char* sensor_decode (uint8_t * *buf*, char * *line*, uint16_t *size*)

Decode sensor data buffer to string.

Parameters

buf	sensor data buffer
line	line buffer for decoded text.
size	maximum size of the line buffer

Returns

a string pointer to line or NULL.

Examples:

xmpl_sensor.c.

3.10.2.3 uint8_t sensor_get (int *idx*, uint8_t * *pdata*)

Query sensor values

Parameters

idx	number of the sensor or ALL_SENSORS (-1) for all
pdata	databuffer to store the sensor data.

Returns

size in bytes of the queried data block (size depends on data availability, e.g. if the sensor has data).

Note: To estimate the max. size for pdata, call sensor_get(-1, NULL). This retrieves the max. block length for all sensors.

Examples:

xmpl_sensor.c.

3.10.2.4 uint8_t sensor_get_error (uint8_t *idx*)

Get last error code of the addressed sensor.

Parameters

idx	number of the sensor.

3.10.2.5 uint8_t sensor_get_id (uint8_t *idx*)

Get sensor ID value for the sensor indexed by idx.

Parameters

idx	number of the sensor.

Examples:

xmpl_sensor.c.

3.10.2.6 char* sensor_get_name (uint8_t *idx*)

get sensor type for the addressed sensor.

Parameters

idx	number of the sensor.

Examples:

xmpl_sensor.c.

3.10.2.7 uint8_t sensor_get_number (void)

Return the number of sensors for this board.

Examples:

xmpl_sensor.c.

3.10.2.8 char* sensor_get_type (uint8_t *idx*)

Get sensor type for the adressed sensor.

Parameters

idx	number of the sensor.

Examples:

xmpl_sensor.c.

3.10.2.9 void sensor_sleep (int *idx*)

Set sensor into sleep mode.

Parameters

idx	number of the sensor or ALL_SENSORS (-1) for all

Examples:

xmpl_sensor.c.

3.10.2.10 void sensor_trigger (int *idx*, bool *one_shot*)

Trigger a sensor measurment.

Parameters

idx	number of the sensor or ALL_SENSORS (-1) for all.
one_shot	if one_shot is true, a single measurement is triggered, otherwise a continous measurement. It depends on the sensor if this parameter is really supported in hardware.

Examples:

xmpl_sensor.c.

3.10.3 Data Structure Documentation

3.10.3.1 struct board_sensor_ctx_t

board defined sensor data structure

Definition at line 76 of file sensor.h.

3.11 I2C Bus Driver

3.11.1 Detailed Description

3.11.2 Functions

3.11.2.1 void i2c_init (uint32_t *freq_Hz*)

initialize I2C bus

Examples:

xmpl_i2c.c, xmpl_isl29020.c, xmpl_lm73.c, xmpl_sensor.c, and xmpl_tsl2550.c.

3.11.2.2 uint8_t i2c_master_writeread (uint8_t *devaddr,* uint8_t * *writebuf,* uint8_t *bytestowrite,* uint8_t * *readbuf,* uint8_t *bytestoread*)

combined read and write of datablocks on the I2C bus.

Examples:

xmpl_i2c.c.

3.11.2.3 uint8_t i2c_probe (uint8_t *devaddr*)

probe the presence of a device on the I2C bus.

Examples:

xmpl_i2c.c.

3.12 One Wire Bus Driver

3.12.1 Detailed Description

- Manual: http://pdfserv.maximintegrated.com/en/an/AN937.pdf

3.12.2 Functions

3.12.2.1 uint8_t ow_byte_read (void)

read byte from one wire bus

Examples:

xmpl_ow.c.

3.12.2.2 void ow_byte_write (uint8_t *byte*)

write byte to one wire bus

Examples:

xmpl_ow.c.

3.12.2.3 bool ow_crc_valid (uint8_t * *pdata,* uint8_t *size*)

calulate and verify iButton CRC

Examples:

xmpl_ow.c.

3.12.2.4 void ow_init (void)

initialize one wire bus

Examples:

xmpl_ow.c.

3.12.2.5 void ow_master_matchrom (ow_serial_t *ser*)

address a device on the one wire bus

3.12.2.6 bool ow_master_searchrom (ow_serial_t * *ser,* bool *first*)

scan one wire bus for presence of devices (tree search)

Examples:

xmpl_ow.c.

3.12.2.7 uint8_t ow_reset (void)

reset one wire bus

Examples:

xmpl_ow.c.

3.13 DS18B20 - One Wire Temperature Sensor.

Data Structures

- struct ds18b20_ctx_t

3.13.1 Detailed Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points.

- Product Page: http://www.maximintegrated.com/en/products/analog/sensors-and-sensor-inter DS18B20.html
- Datasheet: http://pdfserv.maximintegrated.com/en/ds/DS18B20.pdf

3.13.2 Functions

3.13.2.1 static uint8_t ds18b20_command (uint8_t *command,* uint8_t * *buf*) [static]

Send command to 1 wire sensor.

Parameters

command	command code
*buf	argument buffer

Returns

error code

Definition at line 128 of file ds18b20.h.

3.13.2.2 static uint8_t ds18b20_get_val (void * *pctx,* uint8_t * *pdata*) [static]

return temperature value

Definition at line 231 of file ds18b20.h.

3.13.2.3 static uint8_t ds18b20_read_temperature (ow_serial_t *ser,* int16_t * *temp*) [static]

Read the temperature value

Parameters

ser	serial number of the device or 0 if skip addressing (in case if just one sensor is present)
*temp	pointer that holds the singned integer temperature value.

Returns

error code of conversion

Definition at line 180 of file ds18b20.h.

3.13.2.4 static void ds18b20_trigger (void * *pctx,* bool *one_shot*) [static]

start converion not implemented

Definition at line 226 of file ds18b20.h.

3.13.2.5 static uint8_t sensor_create_ds18b20 (void * *pdata,* bool *raw*) [static]

Create an instance of a ds18b20 sensor and initialize the sensor.

Returns

sizeof(ds18b20_ctx_t)

Definition at line 290 of file ds18b20.h.

3.13.3 Data Structure Documentation

3.13.3.1 struct ds18b20_ctx_t

DS18B20 context structure

Definition at line 109 of file ds18b20.h.

3.13.4 Defines

3.13.4.1 #define DS18B20_NB (1)

With the definition of the following macros, multiple ds18b20 sensors can be configured, per default the driver assumes just one (anonymous) sensor.

```
00001 #define DS18B20_NB (3)
00002 #define DS18B20_ADDR { 0xfc000005eab97828, \
00003                        0x90000005eb460e28, \
00004                        0xc5000005ea2dc128 }
```

This macros can either be manually in the apropriate board_xyz.h or they can added in the board.cfg file in the form:

```
00001 sensors: ... ds18b20:0xfc000005eab97828,0x90000005eb460e28,0xc5000005ea2dc128
```

When using the second method, the macros appear in the generated board_cfg.h.

Definition at line 76 of file ds18b20.h.

3.14 HMC5883L - 3-Axis Digital Compass IC

3.14.1 Detailed Description

The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing.

Datasheet: HMC5883L_3-Axis_Digital_Compass_IC.pdf

3.14.2 Functions

3.14.2.1 static uint8_t sensor_create_hmc5883l (void * *pdata*, bool *raw*, uint8_t *addr*) [static]

Create an instance of a hmc5883l sensor and initialize the sensor.

Returns

sizeof(hmc5883l_ctx_t)

Definition at line 122 of file hmc5883l.h.

3.15 ISL 29020 - Light Sensor

3.15.1 Detailed Description

A Low Power, High Sensitivity, Light-to Digital Sensor With I2C Interface.

Datasheet: http://www.intersil.com/content/dam/Intersil/documents/fn65/fn6505.↩
pdf

3.15.2 Defines

3.15.2.1 #define ISL29020_ADDR_0 (0x44)

1st. I2C address option, see macro ISL29020_ADDR in board∗.h for actual configuration.

Definition at line 48 of file isl29020.h.

3.15.2.2 #define ISL29020_ADDR_1 (0x45)

2nd. I2C address option

Definition at line 50 of file isl29020.h.

3.16 LEDPS - Using a LED as Photo Sensor

Data Structures

- struct ledps_ctx_t

3.16.1 Detailed Description

Implementation for using a LED as photo sensor. The LED is connected between two port pins, so that it can be reverse charged.

3.16.2 Functions

3.16.2.1 static uint8_t ledps_get_raw (void * *pctx,* uint8_t * *pdata*) [static]

Retrieve raw value from LED-Photo-Sensor

Definition at line 173 of file ledps.h.

3.16.2.2 static uint8_t ledps_get_val (void * *pctx,* uint8_t * *pdata*) [static]

Retrieve value from LED-Photo-Sensor

Definition at line 159 of file ledps.h.

3.16.2.3 static void ledps_sleep (void * *pctx*) [static]

LED-Photo-Sensor does not support sleep.

Definition at line 189 of file ledps.h.

3.16.2.4 static void ledps_trigger (void * *pctx,* bool *one_shot*) [static]

this function is empty

Definition at line 154 of file ledps.h.

3.16.2.5 static uint16_t sample_port (char *portid,* uint8_t *anode_pin,* uint8_t *cathode_pin*) [static]

```
00001 #define LEDPS_PORTx (1)
00002 #define LEDPS_PORT  ('x')
00003 #define LEDPS_ANODE (y)
00004 #define LEDPS_CATHODE (z)
00005 x: name of PORT, upper case letter A...F
00006 y: port pin for anode 0 ... 7
00007 z: port pin for cathode 0 ... 7
```

Definition at line 105 of file ledps.h.

3.16.2.6 static uint8_t sensor_create_ledps (void * *pdata,* bool *raw,* char *portid,* uint8_t *panode,* uint8_t *pcathode*) [static]

Create an instance of a ledps sensor and initialize the sensor.

Returns

sizeof(ledps_ctx_t)

Definition at line 199 of file ledps.h.

3.16.3 Data Structure Documentation

3.16.3.1 struct ledps_ctx_t

LED context

Definition at line 76 of file ledps.h.

Data Fields

sensor_driver↩ _t	g	
uint8_t	panode	
uint8_t	pcathode	
char	portid	driver structure

3.16.4 Defines

3.16.4.1 #define DO_LED_PS(*port, ddr, pin, anode, cathode, ledps_cnt*)

Value:

```
do { \
            ddr |= (_BV(anode) | _BV(cathode)); \
            port &= ~_BV(anode); \
            port |= _BV(cathode); \
            _delay_us(5); \
            cli(); \
            ddr &= ~_BV(cathode); \
            port &= ~_BV(cathode); \
            ledps_cnt = 0x7fff; \
            do { \
                _delay_us(20); \
            } while ((pin & _BV(cathode)) && --ledps_cnt);\
            sei(); \
            ddr |= (_BV(anode) | _BV(cathode));\
       } while(0)
```

Sampling Macro

Definition at line 55 of file ledps.h.

3.16.4.2 #define LEDPS_DEBUG (0)

Debgging driver with HIF

Definition at line 47 of file ledps.h.

3.17 LM73 - Temperature Sensor

3.17.1 Detailed Description

±1 °C Temperature Sensor with I2C/SMBus Interface

- Product: http://www.ti.com/product/lm73
- Datasheet: http://www.ti.com/lit/ds/symlink/lm73.pdf

3.17.2 Functions

3.17.2.1 static uint8_t sensor_create_lm73 (void * *pdata*, bool *raw*, uint8_t *addr*, int8_t *temp_offset*) [static]

Create an instance of a LM73 sensor and initialize the sensor.

If the parameter pdata != NULL, the sensor is also initialized. In case that the initialization fails, the variable drv.↩ last_error is set to SENSOR_ERR_INIT.

Parameters

pdata	pointer to memory, where the sensor data is stored.
raw	if true the get function will return the register values, otherwise a physical value is returned (temperature in degC a float)
addr	i2c address of the sensor
temp_offset	offset of the measurement value.

Returns

sizeof(lm73_ctx_t)

Examples:

xmpl_lm73.c.

Definition at line 270 of file lm73.h.

3.17.3 Defines

3.17.3.1 #define LM73_ADDR_0 (0x48)

LM3-0, ADDR pin is floating

Definition at line 55 of file lm73.h.

3.17.3.2 #define LM73_ADDR_1 (0x49)

LM3-0, ADDR pin is pulled to GND

Definition at line 56 of file lm73.h.

3.17.3.3 #define LM73_ADDR_2 (0x4A)

LM3-0, ADDR pin is pulled to VDD

Definition at line 57 of file lm73.h.

3.17.3.4 #define LM73_ADDR_3 (0x4C)

LM3-1, ADDR pin is floating

Definition at line 59 of file lm73.h.

3.17.3.5 #define LM73_ADDR_4 (0x4D)

LM3-1, ADDR pin is pulled to GND

Definition at line 60 of file lm73.h.

3.17.3.6 #define LM73_ADDR_5 (0x4E)

LM3-1, ADDR pin is pulled to VDD

Definition at line 61 of file lm73.h.

3.17.3.7 #define LM73_T_BUS_FREE_US (8)

This value is the time between a stop-start cycle of the LM73. It is speced as tBuf = 1.2us min, but on mesh bean it was found that 8us ensure stable behaviour.

Examples:

xmpl_lm73.c.

Definition at line 88 of file lm73.h.

3.18 MCU Temperature Sensor

3.18.1 Detailed Description

Measuring environment temperature with the MCU internal ADC.

3.18.2 Functions

3.18.2.1 static uint8_t mcu_temp_get_raw (void * *pctx,* uint8_t * *pdata*) [static]

not implemented, returns noting

Definition at line 98 of file mcu_temp.h.

3.18.2.2 static uint8_t mcu_temp_get_val (void * *pctx,* uint8_t * *pdata*) [static]

Sensor API Sensor API get functions

Definition at line 69 of file mcu_temp.h.

3.18.2.3 static uint8_t sensor_create_mcu_temp (void * *pdata,* bool *raw*) [static]

Create an instance of a LM73 sensor and initialize the sensor.

If the parameter pdata != NULL, the sensor is also initialized. In case that the initialization fails, the variable drv.↩ last_error is set to SENSOR_ERR_INIT.

Parameters

pdata	pointer to memory, where the sensor data is stored.
raw	if true the get function will return the register values, otherwise a physical value is returned (temperature in degC a float)
addr	i2c address of the sensor
temp_offset	offset of the measurement value.

Returns

sizeof(lm73_ctx_t)

Definition at line 124 of file mcu_temp.h.

3.19 MCU Operating Voltage Sensor

3.19.1 Detailed Description

Measuring operating voltage with the transceiver internal BATMON block.

3.19.2 Functions

3.19.2.1 static uint8_t sensor_create_trxvtg (void * *pdata*) [static]

Create an instance of a trxvtg sensor and initialize the sensor.

Returns

sizeof(trxvtg_ctx_t)

Definition at line 129 of file trxvtg.h.

3.20 TSL2550 - Ambient Light Sensor

3.20.1 Detailed Description

Measuring ambient light intensity.

- Product Page: http://ams.com/eng/Products/Light-Sensors/Ambient-Light-Sensor-ALS/TSL2550
- Datasheet: http://ams.com/eng/content/download/250130/975613/142977

3.20.2 Functions

3.20.2.1 static uint8_t tsl2550_get (uint8_t *adc*) [static]

return raw adc reading (valid flag, log compressed value)

Examples:

xmpl_tsl2550.c.

Definition at line 93 of file tsl2550.h.

3.20.2.2 static uint16_t tsl2550_scale (uint8_t *adc0,* uint8_t *adc1*) [static]

derived from app note dn9b_tsl2550_lux_calculation.pdf

Examples:

xmpl_tsl2550.c.

Definition at line 105 of file tsl2550.h.

3.20.3 Defines

3.20.3.1 #define TSL2550_ADDR (0x39)

the I2C address of the TSL2550

Examples:

xmpl_tsl2550.c.

Definition at line 48 of file tsl2550.h.

3.20.3.2 #define TSL2550_EXT_RANGE (0x1D)

Write command to assert extended range mode

Examples:

xmpl_tsl2550.c.

Definition at line 54 of file tsl2550.h.

3.20.3.3 #define TSL2550_PWR_DOWN (0x00)

Power-down state

Examples:

xmpl_tsl2550.c.

Definition at line 50 of file tsl2550.h.

3.20.3.4 #define TSL2550_RD_ADC0 (0x43)

Read ADC channel 0

Definition at line 58 of file tsl2550.h.

3.20.3.5 #define TSL2550_RD_ADC1 (0x83)

Read ADC channel 1

Definition at line 60 of file tsl2550.h.

3.20.3.6 #define TSL2550_RD_CMD (0x03)

Power-up state/Read command register

Examples:

xmpl_tsl2550.c.

Definition at line 52 of file tsl2550.h.

3.20.3.7 #define TSL2550_STD_RANGE (0x18)

Write command to reset or return to standard range mode

Examples:

xmpl_tsl2550.c.

Definition at line 56 of file tsl2550.h.

3.21 Arduino Radio Functions

Description of the UASP functions.

Data Structures

- struct radio_buffer_t
- class HardwareRadio

3.21.1 Detailed Description

3.21.2 Data Structure Documentation

3.21.2.1 struct radio_buffer_t

Element of a chained list of frame buffers

Definition at line 56 of file HardwareRadio.h.

Data Fields

uint8_t	frm[PHY_MAX↩ _FRAME_SIZE]	array that can store a maximum IEEE 802.15.4 frame
uint8_t	idx	read/write index
uint8_t	len	Length of payload
struct radio_buffer ∗	next	pointer to next list element or NULL if list terminates

3.21.2.2 class HardwareRadio

Hardware Radio class

Definition at line 71 of file HardwareRadio.h.

Public Member Functions

- radio_buffer_t ∗ alloc_buffer (void)
- void free_buffer (radio_buffer_t ∗pbuf)
- HardwareRadio (void)
- void begin (void)
- void begin (uint8_t channel, uint8_t idlestate)
- virtual int available (void)
- virtual int peek (void)
- virtual int read (void)
- virtual void flush (void)
- virtual size_t write (uint8_t)
- void write (char ∗str)
- void write (uint8_t ∗buf, uint8_t size)

Constructor & Destructor Documentation

3.21.2.2.1 HardwareRadio::HardwareRadio (void)

constructor

Member Function Documentation

3.21.2.2.2 radio_buffer_t∗ HardwareRadio::alloc_buffer (void)

Allocate a radio buffer

3.21.2.2.3 virtual int HardwareRadio::available (void) [virtual]

return number of available bytes in current RX buffer

Examples:

Gateway.ino, and RadioUart.ino.

3.21.2.2.4 void HardwareRadio::begin (void)

Starting the hardware radio class with default parameters

Examples:

Gateway.ino, HelloRadio.ino, IoCheck.ino, IoRadio.ino, RadioUart.ino, and Remote.ino.

3.21.2.2.5 void HardwareRadio::begin (uint8_t *channel,* uint8_t *idlestate*)

Starting the hardware radio class with explicit parameters

Parameters

channel	radio channel (11 - 26 for 2.4GHz radios, 0 - 10 for SubGHz radios)
idlestate	default state of the radio, supported values are listed in radio_state_t.

3.21.2.2.6 virtual void HardwareRadio::flush (void) [virtual]

flush TX and RX queues. RX queue data are discarded, TX data is sent.

Examples:

HelloRadio.ino, IoRadio.ino, RadioUart.ino, and Remote.ino.

3.21.2.2.7 void HardwareRadio::free_buffer (radio_buffer_t * *pbuf*)

Free a radio buffer

3.21.2.2.8 virtual int HardwareRadio::peek (void) [virtual]

Returns the next byte (character) of incoming data (RX) without removing it from the internal serial buffer.

Returns

EOF (-1) if no data available, otherwise a value from 0 ... 255

3.21.2.2.9 virtual int HardwareRadio::read (void) [virtual]

Returns the next byte (character) of incoming data (RX)

Returns

EOF (-1) if no data available, otherwise a value from 0 ... 255

Examples:

Gateway.ino, and RadioUart.ino.

3.21.2.2.10 virtual size_t HardwareRadio::write (uint8_t) [virtual]

write a byte to the TX stream

Examples:

HelloRadio.ino, and IoRadio.ino.

3.21.2.2.11 void HardwareRadio::write (char * *str*)

write a string to the TX stream

Parameters

str	\0 terminated string

3.21.2.2.12 void HardwareRadio::write (uint8_t * *buf*, uint8_t *size*)

write a binary buffer (buf, size) to the TX stream

Parameters

buf	pointer to the buffer
size	number of bytes in the buffer.

3.21.3 Defines

3.21.3.1 #define PHY_DEFAULT_CHANNEL (17)

Default radio channel number in 2.4 GHz band

Definition at line 50 of file HardwareRadio.h.

3.21.3.2 #define PHY_MAX_CHANNEL (26)

Maximun radio channel number in 2.4 GHz band

Definition at line 48 of file HardwareRadio.h.

3.21.3.3 #define PHY_MAX_FRAME_SIZE (127)

Maximum size of a IEEE 802.15.4 frame

Definition at line 53 of file HardwareRadio.h.

3.21.3.4 #define PHY_MIN_CHANNEL (11)

Minimun radio channel number in 2.4 GHz band

Definition at line 46 of file HardwareRadio.h.

4 Data Structure Documentation

4.1 NWK_DataInd_t Struct Reference

structure for receiving frames

Data Fields

- uint16_t srcAddr
- uint8_t dstEndpoint
- uint8_t srcEndpoint
- NWK_Ind_Opt_t options
- uint8_t * data
- uint8_t size
- uint8_t lqi
- uint8_t rssi

4.1.1 Detailed Description

Definition at line 192 of file lw_mesh.h.

4.1.2 Field Documentation

4.1.2.1 uint8_t* NWK_DataInd_t::data

pointer to the payload data

Definition at line 199 of file lw_mesh.h.

4.1.2.2 uint8_t NWK_DataInd_t::dstEndpoint

destination endpoint (local)

Definition at line 195 of file lw_mesh.h.

4.1.2.3 uint8_t NWK_DataInd_t::lqi

LQI value reported by usr_radio_receive_frame

Definition at line 201 of file lw_mesh.h.

4.1.2.4 NWK_Ind_Opt_t NWK_DataInd_t::options

data request options, see NWK_Ind_Opt_t

Definition at line 197 of file lw_mesh.h.

4.1.2.5 uint8_t NWK_DataInd_t::rssi

RSSI value reported by usr_radio_receive_frame

Definition at line 203 of file lw_mesh.h.

4.1.2.6 uint8_t NWK_DataInd_t::size

size of the payload data

Definition at line 200 of file lw_mesh.h.

4.1.2.7 uint16_t NWK_DataInd_t::srcAddr

short source address

Definition at line 194 of file lw_mesh.h.

4.1.2.8 uint8_t NWK_DataInd_t::srcEndpoint

source endpoint (remote)

Definition at line 196 of file lw_mesh.h.

4.2 NWK_DataReq_t Struct Reference

structure for sending frames

Data Fields

- uint16_t dstAddr
- uint8_t dstEndpoint
- uint8_t srcEndpoint
- NWK_Opt_t options
- uint8_t * data
- uint8_t size
- void(* confirm)(struct NWK_DataReq_tag *)
- NWK_Stat_t status
- uint8_t control

4.2.1 Detailed Description

Definition at line 174 of file lw_mesh.h.

4.2.2 Field Documentation

4.2.2.1 void(* NWK_DataReq_t::confirm) (struct NWK_DataReq_tag *)

pointer to confirmation callback function

Definition at line 182 of file lw_mesh.h.

4.2.2.2 uint8_t NWK_DataReq_t::control

filled by stack, value of ack ctrl

Definition at line 185 of file lw_mesh.h.

4.2.2.3 uint8_t* NWK_DataReq_t::data

pointer to the payload data

Definition at line 180 of file lw_mesh.h.

4.2.2.4 uint16_t NWK_DataReq_t::dstAddr

short destination address

Definition at line 176 of file lw_mesh.h.

4.2.2.5 uint8_t NWK_DataReq_t::dstEndpoint

destination endpoint (remote)

Definition at line 177 of file lw_mesh.h.

4.2.2.6 NWK_Opt_t NWK_DataReq_t::options

data request options, see NWK_Opt_t

Definition at line 179 of file lw_mesh.h.

4.2.2.7 uint8_t NWK_DataReq_t::size

size of the payload data

Definition at line 181 of file lw_mesh.h.

4.2.2.8 uint8_t NWK_DataReq_t::srcEndpoint

source endpoint (local)

Definition at line 178 of file lw_mesh.h.

4.2.2.9 NWK_Stat_t NWK_DataReq_t::status

filled by stack, status of this Req

Definition at line 184 of file lw_mesh.h.

4.3 NWK_FrameFormat_t Struct Reference

general lightweight mesh frame format

Data Fields

- uint16_t m_fctl
- uint8_t m_seq
- uint16_t m_pid
- uint16_t m_dstAddr
- uint16_t m_srcAddr
- uint8_t lw_fctl
- uint8_t lw_seq
- uint16_t lw_srcAddr
- uint16_t lw_dstAddr
- uint8_t lw_endpts
- uint8_t lw_payload [NWK_MAX_PAYLOAD_SIZE]
- uint16_t m_crc

4.3.1 Detailed Description

Definition at line 211 of file lw_mesh.h.

4.3.2 Field Documentation

4.3.2.1 uint16_t NWK_FrameFormat_t::lw_dstAddr

lightweight mesh destination address

Definition at line 221 of file lw_mesh.h.

4.3.2.2 uint8_t NWK_FrameFormat_t::lw_endpts

lightweight mesh src and dest endpoints

Definition at line 222 of file lw_mesh.h.

4.3.2.3 uint8_t NWK_FrameFormat_t::lw_fctl

lightweight mesh frame control field

Definition at line 218 of file lw_mesh.h.

4.3.2.4 uint8_t NWK_FrameFormat_t::lw_payload[NWK_MAX_PAYLOAD_SIZE]

application payload

Definition at line 223 of file lw_mesh.h.

4.3.2.5 uint8_t NWK_FrameFormat_t::lw_seq

lightweight mesh sequence number

Definition at line 219 of file lw_mesh.h.

4.3.2.6 uint16_t NWK_FrameFormat_t::lw_srcAddr

lightweight mesh source address

Definition at line 220 of file lw_mesh.h.

4.3.2.7 uint16_t NWK_FrameFormat_t::m_crc

MAC check sum

Definition at line 225 of file lw_mesh.h.

4.3.2.8 uint16_t NWK_FrameFormat_t::m_dstAddr

MAC destination address

Definition at line 215 of file lw_mesh.h.

4.3.2.9 uint16_t NWK_FrameFormat_t::m_fctl

MAC frame control field

Definition at line 212 of file lw_mesh.h.

4.3.2.10 uint16_t NWK_FrameFormat_t::m_pid

MAC PAN ID

Definition at line 214 of file lw_mesh.h.

4.3.2.11 uint8_t NWK_FrameFormat_t::m_seq

MAC sequence number

Definition at line 213 of file lw_mesh.h.

4.3.2.12 uint16_t NWK_FrameFormat_t::m_srcAddr

MAC source address

Definition at line 216 of file lw_mesh.h.

4.4 NWK_Tasklist_t Struct Reference

structure for task list to propagate NWK_DataReq_t and NWK_DataInd_t to lw_mesh_task_handler

Data Fields

- NWK_FrameFormat_t * frame

4.4.1 Detailed Description

Definition at line 241 of file lw_mesh.h.

4.4.2 Field Documentation

4.4.2.1 NWK_FrameFormat_t* NWK_Tasklist_t::frame

Pointer to frame

Definition at line 245 of file lw_mesh.h.

4.5 p2p_hdr_t Struct Reference

Data Fields

- uint16_t fcf
- uint8_t seq
- uint16_t pan
- uint16_t dst
- uint16_t src
- uint8_t cmd

4.5.1 Detailed Description

Frame header structure for P2P applications. This header is formated as a IEEE 802.15.4 frame header with short addressing mode for src and dst and pan comression set to on.

Definition at line 94 of file p2p_protocol.h.

4.5.2 Field Documentation

4.5.2.1 uint8_t p2p_hdr_t::cmd

the command code that identifies the frame, see p2p_cmdcode_t

Definition at line 101 of file p2p_protocol.h.

4.5.2.2 uint16_t p2p_hdr_t::dst

destination short address

Definition at line 99 of file p2p_protocol.h.

4.5.2.3 uint16_t p2p_hdr_t::fcf

frame control field

Definition at line 96 of file p2p_protocol.h.

4.5.2.4 uint16_t p2p_hdr_t::pan

destination pan id

Definition at line 98 of file p2p_protocol.h.

4.5.2.5 uint8_t p2p_hdr_t::seq

sequence number

Definition at line 97 of file p2p_protocol.h.

4.5.2.6 uint16_t p2p_hdr_t::src

source short address

Definition at line 100 of file p2p_protocol.h.

4.6 p2p_jump_bootl_t Struct Reference

4.6.1 Detailed Description

Frame structure for P2P_JUMP_BOOTL.

Definition at line 158 of file p2p_protocol.h.

4.7 p2p_ping_cnf_t Struct Reference

Data Fields

- p2p_status_t status
- p2p_error_t errno
- uint8_t version
- uint16_t crc
- uint8_t appname [8]
- uint8_t boardname [16]

4.7.1 Detailed Description

Frame structure for P2P_PING_CNF.

Definition at line 146 of file p2p_protocol.h.

4.7.2 Field Documentation

4.7.2.1 uint8_t p2p_ping_cnf_t::appname[8]

application identification string

Definition at line 153 of file p2p_protocol.h.

4.7.2.2 uint8_t p2p_ping_cnf_t::boardname[16]

board identification string

Definition at line 154 of file p2p_protocol.h.

4.7.2.3 uint16_t p2p_ping_cnf_t::crc

checksum of received data (only used from WIBO)

Definition at line 152 of file p2p_protocol.h.

4.7.2.4 p2p_error_t p2p_ping_cnf_t::errno

current error code

Definition at line 150 of file p2p_protocol.h.

4.7.2.5 p2p_status_t p2p_ping_cnf_t::status

application status

Definition at line 149 of file p2p_protocol.h.

4.7.2.6 uint8_t p2p_ping_cnf_t::version

software version

Definition at line 151 of file p2p_protocol.h.

4.8 p2p_ping_req_t Struct Reference

4.8.1 Detailed Description

Frame structure for P2P_PING_REQ.

Definition at line 139 of file p2p_protocol.h.

4.9 p2p_sensor_caption_t Struct Reference

Data Fields

- p2p_hdr_t hdr
- uint8_t caption [ ]

4.9.1 Detailed Description

Frame structure for P2P_SENSOR_CAPTION.

Definition at line 231 of file p2p_protocol.h.

4.9.2 Field Documentation

4.9.2.1 uint8_t p2p_sensor_caption_t::caption[ ]

NULL terminated string

Definition at line 234 of file p2p_protocol.h.

4.9.2.2 p2p_hdr_t p2p_sensor_caption_t::hdr

p2p frame header

Definition at line 233 of file p2p_protocol.h.

4.10 p2p_sensor_data_t Struct Reference

Data Fields

- p2p_hdr_t hdr

4.10.1 Detailed Description

Frame structure for P2P_SENSOR_DATA.

Definition at line 224 of file p2p_protocol.h.

4.10.2 Field Documentation

4.10.2.1 p2p_hdr_t p2p_sensor_data_t::hdr

p2p frame header

Definition at line 226 of file p2p_protocol.h.

4.11 p2p_wibo_addr_t Struct Reference

4.11.1 Detailed Description

Frame structure for P2P_WIBO_ADDR.

Definition at line 191 of file p2p_protocol.h.

4.12 p2p_wibo_bootlup_t Struct Reference

4.12.1 Detailed Description

Frame structure for P2P_WIBO_BOOTLUP.

Definition at line 204 of file p2p_protocol.h.

4.13 p2p_wibo_data_t Struct Reference

Data Fields

- uint8_t dsize
- uint8_t data [ ]

4.13.1 Detailed Description

Frame structure for P2P_WIBO_DATA.

Definition at line 164 of file p2p_protocol.h.

4.13.2 Field Documentation

4.13.2.1 uint8_t p2p_wibo_data_t::data[ ]

data container

Definition at line 167 of file p2p_protocol.h.

4.13.2.2 uint8_t p2p_wibo_data_t::dsize

size of data packet

Definition at line 166 of file p2p_protocol.h.

4.14 p2p_wibo_exit_t Struct Reference

4.14.1 Detailed Description

Frame structure for P2P_WIBO_EXIT.

Definition at line 198 of file p2p_protocol.h.

4.15 p2p_wibo_finish_t Struct Reference

4.15.1 Detailed Description

Frame structure for P2P_WIBO_FINISH.

Definition at line 171 of file p2p_protocol.h.

4.16 p2p_wibo_reset_t Struct Reference

4.16.1 Detailed Description

Frame structure for P2P_WIBO_RESET.

Definition at line 178 of file p2p_protocol.h.

4.17 p2p_wibo_target_t Struct Reference

Data Fields

- uint8_t targmem

4.17.1 Detailed Description

Frame structure for P2P_WIBO_TARGET.

Definition at line 184 of file p2p_protocol.h.

4.17.2 Field Documentation

4.17.2.1 uint8_t p2p_wibo_target_t::targmem

target memory: EEPROM or FLASH or LOCKBITS

Definition at line 187 of file p2p_protocol.h.

4.18 p2p_wuart_data_t Struct Reference

Data Fields

- p2p_hdr_t hdr
- uint8_t mode

4.18.1 Detailed Description

Frame structure for P2P_WUART_DATA.

Definition at line 217 of file p2p_protocol.h.

4.18.2 Field Documentation

4.18.2.1 p2p_hdr_t p2p_wuart_data_t::hdr

p2p frame header

Definition at line 219 of file p2p_protocol.h.

4.18.2.2 uint8_t p2p_wuart_data_t::mode

reserved for future extension

Definition at line 220 of file p2p_protocol.h.

4.19 p2p_xmpl_led_t Struct Reference

4.19.1 Detailed Description

Frame structure for P2P_XMPL_LED.

Definition at line 210 of file p2p_protocol.h.

4.20 sensor_driver_t Struct Reference

4.20.1 Detailed Description

sensor driver structure

Definition at line 59 of file sensor_defs.h.

4.21 sensor_light_t Struct Reference

4.21.1 Detailed Description

sensor value for light

Definition at line 77 of file sensor_defs.h.

4.22 sensor_magnetic_t Struct Reference

4.22.1 Detailed Description

sensor value for 3-axis magnetic, typical unit is Gauss

Definition at line 85 of file sensor_defs.h.

4.23 sensor_raw_t Struct Reference

4.23.1 Detailed Description

sensor value for raw value

Definition at line 104 of file sensor_defs.h.

4.24 sensor_temperature_t Struct Reference

4.24.1 Detailed Description

sensor value for temperature

Definition at line 69 of file sensor_defs.h.

4.25 sensor_voltage_t Struct Reference

4.25.1 Detailed Description

sensor value for light

Definition at line 95 of file sensor_defs.h.

5 Example Documentation

5.1 Gateway.ino

This sketch implements a COSM Gateway.

```
/*
  Cosm temperature logger

  This sketch is derived from PachubeClientString Example bundled with
  Arduino. It receives a character line from temperature sensor and feeds
  it to cosm.

  Hardware:
   - Radiofaro + DFRobot Ethernet Shield attached

 */

#include <SPI.h>
#include <Ethernet.h>
#include "HardwareRadio.h"


#define APIKEY         "The quick brown fox" // replace your Pachube api key here
#define FEEDID         123456 // replace your feed ID
#define USERAGENT      "Radiofaro" // user agent is the project name


// assign a MAC address for the ethernet controller.
// fill in your address here:
  byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

// fill in an available IP address on your network here,
// for manual configuration:
IPAddress ip(10,0,1,20);

// initialize the library instance:
EthernetClient client;

// if you don't want to use DNS (and reduce your sketch size)
// use the numeric IP instead of the name for the server:
IPAddress server(216,52,233,121);      // numeric IP for api.pachube.com
//char server[] = "api.pachube.com";   // name address for pachube API

unsigned long lastConnectionTime = 0;          // last time you connected to the server, in milliseconds
boolean lastConnected = false;                 // state of the connection last time through the main loop
const unsigned long postingInterval = 20*1000;  //delay between updates to pachube.com

char ln[80]; /* max line length */
byte idx = 0;
byte lncnt = 0;
boolean hasData=0;
```

```
String dataString = "";

String addr_whitelist[] = {"0xFECA", "0xAFFE"};

void setup() {
  Radio.begin();
  Serial.begin(9600);

  // give the ethernet module time to boot up:
  delay(1000);
  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // DHCP failed, so use a fixed IP address:
    Ethernet.begin(mac, ip);
  }
}

boolean parse(char *ln)
{
    byte i=0;
    char *sarr[8];
    char *s = ln;
    boolean valid=0;

    Serial.print("Line In: ");
    Serial.print(ln); // line break already in line

    /* tokenize to all space separated */
    do{
      if((*s==',') || (*s == '=')) *s=' ';
    }while(*s++);

    s=sarr[i++]=strtok(ln, " ");
    dataString="";
    do{
      s=sarr[i++]=strtok(NULL, " ");
    }while( (NULL != s) && (i<sizeof(sarr)/sizeof(sarr[0])) );

    if(sarr[2][0]=='T' && sarr[4][0]=='V'){

        s=sarr[3];
        while(*s && (*s>='0' && *s<='9' || *s=='.' || *s=='-') ){
          s++;
        };

        if(*s){ /* did not reach NULL terminator */
          /* N.a.N. */
        } else {
          dataString += "temp,";
          dataString += sarr[3];
          dataString += "\nvolt,";
          dataString += sarr[5];
          valid = 1;
        }
    }

    if(!valid){
       Serial.println("Parsing error");
    } else {
      Serial.print("Parsed: ");
      Serial.println(dataString);
    }
    return valid;
}

void loop() {

  if (Radio.available() > 0)
  {
    char c = Radio.read();

    if(c == '\0'){ /* unexpected input, restart */
      idx = 0;
    } else {
      ln[idx] = c;
      idx++;
    }
    ln[idx] = '\0';

    if( ('\n' == c) || ('\r' == c) ) { /* line completed */
      hasData = parse(ln);

      lncnt++;
      idx = 0;
    }
  }
```

```
  // if there's incoming data from the net connection.
  // send it out the serial port.  This is for debugging
  // purposes only:
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  // if there's no net connection, but there was one last time
  // through the loop, then stop the client:
  if (!client.connected() && lastConnected) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
  }

  // if you're not connected, and ten seconds have passed since
  // your last connection, then connect again and send data:
  if(!client.connected() && (millis() - lastConnectionTime > postingInterval) && hasData>0) {
    Serial.print("Pachube: ");
    Serial.println(dataString);

    sendData(dataString);
    hasData=0;
  }
  // store the state of the connection for next time through
  // the loop:
  lastConnected = client.connected();
}

// this method makes a HTTP connection to the server:
void sendData(String thisData) {
  // if there's a successful connection:
  if (client.connect(server, 80)) {
    Serial.println("connecting...");
    // send the HTTP PUT request:
    client.print("PUT /v2/feeds/");
    client.print(FEEDID);
    client.println(".csv HTTP/1.1");
    client.println("Host: api.pachube.com");
    client.print("X-pachubeApiKey: ");
    client.println(APIKEY);
    client.print("User-Agent: ");
    client.println(USERAGENT);
    client.print("Content-Length: ");
    client.println(thisData.length());

    // last pieces of the HTTP PUT request:
    client.println("Content-Type: text/csv");
    client.println("Connection: close");
    client.println();

    // here's the actual content of the PUT request:
    client.println(thisData);
  }
  else {
    // if you couldn't make a connection:
    Serial.println("connection failed");
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
  }
  // note the time that the connection was made or attempted:
  lastConnectionTime = millis();
}
```

5.2 HelloRadio.ino

This sketch sends frames with a period of 500ms.

```
/* $Id$ */
#include "HardwareRadio.h"

#define REG(name, reg) do{uint8_t _b_ = reg; Serial.print(name);Serial.print(" : ");
       Serial.println(_b_,HEX);}while(0)

unsigned long tx_time;
int cnt = 0;
void setup() {

    /* operating on channel 17, not receiving when idle. */
```

```
    Radio.begin(17, STATE_OFF);
    Serial.begin(57600);
    Serial.println("HelloRadio V$Release$");
}

void loop() {

    if (millis() > tx_time){
        tx_time = millis() + 500;
        // write string
        Radio.write("cnt: ");
        // write integer
        Radio.print(cnt);
        // write byte
        Radio.write('\n');
        Radio.flush();
        Serial.println(cnt);
        cnt ++;
    }

}
```

5.3 IoCheck.ino

This is sketch performs a simple GPIO check, no radio functions included.

```
/* $Id$ */
#include "HardwareRadio.h"

bool DO_ECHO = false;

void setup()
{

    Serial.begin(57600);
    // flush input
    while(Serial.available())
    {
        Serial.read();
    }
    Radio.begin();
    Serial.println("Sketch IoCheck\n\r"
                   "Type help for a list of commands");
}

void loop()
{
    static uint8_t val = 0;
    static char line[32], *pcmd;

    if (DO_ECHO)
    {
        Serial.print("\n\rIoCheck>");
    }
    /* check serial IO */
    pcmd = readline(line, sizeof(line));

    if(pcmd != NULL)
    {
        if (DO_ECHO)
        {
            Serial.println(pcmd);
        }
        process_command(pcmd);
    }
}

// ================================================================
// parse command and dispatch to execution function
void process_command(char *pcmd)
{
    char *argv[8], *p;
    int argc = 0, i;
    bool newarg;

    p = pcmd;
    argv[argc++] = p;
    newarg = false;

    while(*p != 0)
    {
```

```
        if (*p == ' ')
        {
            *p = 0;
            newarg = true;
        }
        else
        {
            if (newarg == true && argc < 8)
            {
                argv[argc++] = p;
                newarg = false;
            }
        }
        p++;
    }
    #if 0
    // dump result of line parsing
    for (i=0; i<argc; i++)
    {
        Serial.print(" - ");
        Serial.print(i);
        Serial.print(":");
        Serial.println(argv[i]);
    }
    #endif

    if (strncasecmp("pin", argv[0], 3) == 0)
    {
        execute_pin_command(argv + 1, argc - 1);
    }
    else if (strncasecmp("help", argv[0], 4) == 0)
    {
        execute_print_help();
    }
    else if (strncasecmp("echo", argv[0], 4) == 0)
    {
        DO_ECHO = atoi(argv[1]) ? true : false;
        Serial.print("echo is now ");
        Serial.println(DO_ECHO ? "ON": "OFF");
    }
    else if (strncasecmp("jtag", argv[0], 4) == 0)
    {
        execute_set_jtag(atoi(argv[1]) ? true : false);
    }
    else if (strncasecmp("sleep", argv[0], 4) == 0)
    {
        execute_set_sleep();
    }
    else if (strncasecmp("tx", argv[0], 2) == 0)
    {
        execute_transmit();
    }
    else if (strncasecmp("rx", argv[0], 2) == 0)
    {
        execute_receive(atoi(argv[1]));
    }
}

void execute_print_help(void)
{
    Serial.print("\n\r"
        " pin <id> <dir> <val> : set pin (id is e.g. 'd0' or 'a0', dir: 0=input, 1=output)\n\r"
        " echo <bool>          : switch echo ON/OFF\n\r"
        " jtag <bool>          : switch jtag ON/OFF\n\r"
        " sleep                : set MCU and TRX to sleep\n\r"
        " tx                   : transmit frame\n\r"
        " rx <tmo>             : receive frame (tmo to wait for frame in ms)\n\r"
        " help                 : print help\n\r");
}
```

5.4 loRadio.ino

This sketch reads A0 and A1 each 500ms, the results are printed to serial terminal and to the Radio.

```
/* $Id$ */
#include <stdio.h>
#include "HardwareRadio.h"

#define REG(name, reg) do{uint8_t _b_ = reg; Serial.print(name);Serial.print(" : ");
       Serial.println(_b_,HEX);}while(0)

unsigned long tx_time;
int cnt = 0;
```

```
void setup() {

    /* operating on channel 17, not receiving when idle. */
    Radio.begin(17, STATE_OFF);
    Serial.begin(57600);
    Serial.println("RadioIo");
}

void loop() {
int a0, a1;
char buf[64];

    if (millis() > tx_time)
    {
       a0 = analogRead(0);
       a1 = analogRead(1);
       snprintf(buf, 64, " a0: %d, a1: %d\r\n", a0, a1);

       // sent wireless
       Radio.write("cnt: ");
       Radio.print(cnt);
       Radio.write(buf);
       Radio.flush();

       // local echo of results
       Serial.print("cnt:");
       Serial.print(cnt);
       Serial.print(buf);

       tx_time = millis() + 500;
       cnt ++;
    }

}
```

5.5 RadioUart.ino

This sketch implements a wireless UART application.

```
/* $Id$ */
/* RadioFaro wirless/serial bridge. */
#include "HardwareRadio.h"

/* data captured from UART */
char serial_inbyte = 0;

/* data received from Radio */
char serial_outbyte = 0;

/* next time the radio buffer will flushed */
unsigned long tx_time;

void setup() {
    Radio.begin();
    Serial.begin(57600);
    tx_time = 0;
    Serial.println("RadioUart.ino");
}

void loop() {

    if ((Serial.available() > 0))
    {
        serial_inbyte = Serial.read();
        Radio.print(serial_inbyte);
    }

    if (millis() > tx_time){
        tx_time = millis() + 25;
        Radio.flush();
    }

    if (Radio.available() > 0)
    {
        serial_outbyte = Radio.read();
        Serial.print(serial_outbyte);
    }
}
```

5.6 Remote.ino

This sketch implements a COSM remote sensor, that measures temperature and VCC. The board sleeps during the idle period and is woken up by the watch dog.

```
/*
  Cosm temperature logger

  Remote node application

  Hardware:
   - Radiofaro on battery power, temperature sensor internal
*/

#include <avr/wdt.h>
#include <avr/sleep.h>
#include "HardwareRadio.h"

/* === globals ============================================================= */
static volatile uint8_t adcfinished = 0;
static int8_t adc_offset = 0;

/* === functions =========================================================== */

/*
 * \brief Setup watchdog to serve as application timer
 *
 */
static inline void wdt_timersetup(uint8_t timeout)
{
    WDTCSR = (1 << WDCE) | (1 << WDE); /* Enable configuration change */
    WDTCSR = (1 << WDIF) | (1 << WDIE) | /* Enable Watchdog Interrupt Mode */
    (timeout & 0x08 ? _WD_PS3_MASK : 0x00) | (timeout & 0x07);
}

/*
 * \brief Watchdog ISR, used as application timer
 *
 */
ISR(WDT_vect, ISR_NOBLOCK)
{
  /* do nothing, just wake up MCU */
}

ISR(ADC_vect, ISR_BLOCK)
{
    adcfinished = 1;
}

/*
 * \brief Trigger sleep based ADC measurement
 * Function is blocking until flag "adcfinished" is set by ISR
 *
 * @return ADC register value
 */
static inline int16_t adc_measure(void)
{
    set_sleep_mode(SLEEP_MODE_ADC);
    /* dummy cycle */
    adcfinished = 0;
    do
    {
        sleep_mode();
        /* sleep, wake up by ADC IRQ */

        /* check here for ADC IRQ because sleep mode could wake up from
         * another source too
         */
    } while (0 == adcfinished); /* set by ISR */
    return ADC ;
}

float measure_tmcu(void)
{
    int32_t val = 0;
    uint8_t nbavg = 5;

    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1); /* PS 64 */
    ADCSRB = (1 << MUX5);

    ADMUX = (1 << REFS1) | (1 << REFS0) | (1 << MUX3) | (1 << MUX0); /* reference: 1.6V, input Temp Sensor
      */
    _delay_us(200); /* some time to settle */

    ADCSRA |= (1 << ADIF); /* clear flag */
```

```
    ADCSRA |= (1 << ADIE);

    /* dummy cycle after REF change (suggested by datasheet) */
    adc_measure();

    _delay_us(100); /* some time to settle */

    for(uint8_t i=0;i<nbavg;i++){
        val += adc_measure() - adc_offset;
    }

    ADCSRA &= ~((1 << ADEN) | (1 << ADIE));

    return (1.13 * val / (float)nbavg - 272.8);
}

int8_t measure_adc_offset(void)
{
    uint16_t val;

    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1); /* PS 64 */
    ADCSRB = 0;
    ADMUX = (1 << REFS1) | (1 << REFS0) | (1 << MUX3); /* reference: 1.6V, differential ADC0-ADC0 10x */

    _delay_us(200); /* some time to settle */

    ADCSRA |= (1 << ADIF); /* clear flag */
    ADCSRA |= (1 << ADIE);

    /* dummy cycle after REF change (suggested by datasheet) */
    adc_measure();

    _delay_us(100); /* some time to settle */

    val = adc_measure();

    ADCSRA &= ~((1 << ADEN) | (1 << ADIE));

    return (val);
}

/*
 * \brief Measure internal voltage of ATmega128RFA1
 *
 * Cannot be measured via ADC, use Batmon of TRX part
 *
 */
float measure_vmcu(void)
{
    uint16_t v; /* voltage in [mV] */
    uint8_t vth;

    for(vth=0;vth<32;vth++){
        BATMON = vth & 0x1F;
        BATMON = vth & 0x1F;

        if(0 == (BATMON & (1<<BATMON_OK))){
            break;
        }
    }

    if(vth & 0x10){
        v = 2550 + 75*(vth&0x0F); /* high range */
    }else{
        v = 1700 + 50*(vth&0x0F); /* low range */
    }

    return v / 1000.0;
}

void setup()
{
  /* init all unused pins */
  DDRB = 0x00;  /* as input */
  PORTB = 0xFF; /* pullups */
  DDRD = 0x00;  /* as input */
  PORTD = 0xFF; /* pullups */
  DDRE = 0x00;  /* as input */
  PORTE = 0xFF; /* pullups */
  DDRF = 0x00;  /* as input */
  PORTF = 0xFF; /* pullups */
  DDRG = 0x00;  /* as input */
  PORTG = 0xFF; /* pullups */

  DIDR0 = 0xFF; /* disable all ADC inputs */

  /* we are using async Tx only, never receive
```

```
   * anything. Therefore we choose sleep as idle state
   */
  Radio.begin(17, STATE_SLEEP);

  adc_offset = measure_adc_offset();

  wdt_timersetup(WDTO_8S);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
}

void loop()
{
  float tmcu, vmcu;

  tmcu = measure_tmcu();
  vmcu = measure_vmcu();

  Radio.print("ADDR=");
  Radio.print(Radio.nc.short_addr);

  Radio.print(", T=");
  Radio.print(tmcu);

  Radio.print(", V=");
  Radio.print(vmcu);

  Radio.print('\n');

  Radio.flush();
  while(Radio.tx_in_progress);

  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_mode();
}
```

5.7 xmpl_dbg.c

```
/* Copyright (c) 2009 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for use of the DBG_XXX macros */

#include "board.h"
#include "xmpl.h"


int main(void)
{

    mcu_init();
    DBG_INIT();
    /* the debug port pin should now have the level 0 */

    DBG_SET();
    /* the debug port pin should now have the level 1 */
```

```
    DBG_CLR();
    /* the debug port pin should now have again the level 0 */

    while(1)
    {
        DBG_TOGGLE();
        DELAY_MS(10);
        /* now 1-0-1 sequence should be observed with a period of 10ms */
    }
}
/* EOF */
```

5.8 xmpl_hif.c

The programm dumps at first a view text lines to the host interface, then waiting for input data. All characters received are echoed, except for '\n' and '\r' which where treated as line break, printing out a prompt with the current line number in square brackets.

This is an example screenshot of the terminal window, with the output of tree command, redirected to the HIF.

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for use of the HIF functions */

#include "board.h"
#include "hif.h"
#include "xmpl.h"

#define PROMPT() PRINTF("\n\ruracoli[%02d]> ",lc++)

int main(void)
{
 int inchar;
 const uint32_t br = HIF_DEFAULT_BAUDRATE;
 uint8_t lc = 0;
 uint8_t msg[] = { 0x57, 0x65, 0x6c, 0x63, 0x6f, 0x6d, 0x65, 0x20,
                   0x69, 0x6e, 0x20, 0x74, 0x68, 0x65, 0x20, 0x77,
                   0x6f, 0x72, 0x6c, 0x64, 0x20, 0x6f, 0x66, 0x20,
                   0xb5, 0x72, 0x61, 0x63, 0x6f, 0x6c, 0x69, 0x21,
                   '\n','\r',0x00};

    mcu_init();
    /* setting up UART and adjusting the baudrate */
    hif_init(br);

    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
     * been established within the host OS.
     */
    do
    {
```

```
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif

    /* using the basic hif_xxx functions */
    hif_printf(FLASH_STRING("\n\rHIF Example : %s : %ld bit/s\n\r"),BOARD_NAME,br);
    hif_echo(FLASH_STRING("$Revision$\n\r"));

    /* this macro is equivalent to hif_printf(FLASH_STRING(...),...) */
    PRINTF("File: %s:%d\n\r",__FILE__,__LINE__);

    /* this function outputs an array transparently  */
    hif_put_blk (msg, sizeof(msg));

    /* there is also an optimized hexdump function for byte arrays */
    DUMP(sizeof(msg),msg);

    /* starting the main loop, prompting and counting the line numbers */
    PROMPT();
    while(1)
    {
        inchar = hif_getc();
        if (EOF != inchar)
        {
            if (inchar == '\r' || inchar == '\n')
            {
                PROMPT();
            }
            else
            {
                hif_putc(inchar);
            }
        }
    }
}
/* EOF */
```

5.9 xmpl_hif_echo.c

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example that implements HIF echo, usefull to test the HIF troughput */

#include "board.h"
#include "hif.h"
#include "xmpl.h"

int main(void)
{
 int inchar;
 uint32_t br = HIF_DEFAULT_BAUDRATE;

 uint32_t txcnt, rxcnt;
 uint8_t state;
```

```
const char volatile INFO[] = "info";

    mcu_init();

    /* setting up UART and adjusting the baudrate */
    hif_init(br);
    LED_INIT();
    txcnt = 0;
    rxcnt = 0;
    state = 0;
    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
     * been established within the host OS.
     */
    do
    {
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif

    /* using the basic hif_xxx functions */
    hif_printf(FLASH_STRING("\n\rHIF Echo : %s : %ld bit/s\n\r"),BOARD_NAME,br);
    hif_echo(FLASH_STRING("$Revision$\n\r"));

    while(1)
    {
        inchar = hif_getc();
        rxcnt ++;

        if (EOF != inchar)
        {
            hif_putc(inchar);
            txcnt ++;

            if (inchar == INFO[state])
            {
                state += 1;
            }
            else
            {
                state = 0;
            }
            if (state == 4)
            {
                PRINTF("\n\rECHO rx=%ld tx=%ld\n\r", rxcnt, txcnt);
                state = 0;
                txcnt = 0;
                rxcnt = 0;
            }
        }
    }
}
/* EOF */
```

5.10 xmpl_i2c.c

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
```

```
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for use of the I2C functions */
#include <stdlib.h>
#include "board.h"
#include "ioutil.h"
#include "i2c.h"
#include "hif.h"
#include "xmpl.h"


/* === macros ============================================================== */
#define PROMPT() PRINT("\n\ri2c>>> ")
#define LINE_SIZE (80)
#define MAX_ARGS  (16)
#define EOL "\n"

/* === globals ============================================================= */
char *argv[MAX_ARGS];
int argc;

/* === prototypes ========================================================== */
bool process_input(int c);
void process_commands(char **argv, int argc);

/* === implementation ====================================================== */
int main(void)
{
 const uint32_t br = HIF_DEFAULT_BAUDRATE;
 uint8_t addr, ret;
 int chr;

    mcu_init();
    /* setting up UART and adjusting the baudrate */
    hif_init(br);
    LED_INIT();
    LED_SET(0);
    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
     * been established within the host OS.
     */
    do
    {
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif

    i2c_init( 4000000UL );

    /* using the basic hif_xxx functions */
    PRINTF("\n\rI2C Example : %s : %ld bit/s\n\r", BOARD_NAME, br);

    PRINT(EOL"i2c bus scan:"EOL);
    for (addr=0; addr<128; addr++)
    {
        ret = i2c_probe(addr<<1);
        if (ret)
        {
            PRINTF(" a=0x%02x, rv=0x%02x, OK"EOL, addr<<1, ret);
            LED_TOGGLE(1);
        }
        WAIT_MS(1);
        LED_TOGGLE(0);
    }



    PROMPT();
    do
    {
        chr = hif_getc();
        if (process_input(chr))
        {
            if (argc)
            {
                process_commands(argv, argc);
            }
            PROMPT();
        }
```

```
    }
    while (1);

}

bool process_input(int c)
{
    static char line[LINE_SIZE];
    static int idx;

    bool rv = false;
    if (c != EOF)
    {
        if (c == '\n' || c == '\r')
        {
            line[idx] = 0;
            idx = 0;
            hif_puts(EOL);
            argc = hif_split_args(line, MAX_ARGS, argv);
            rv = true;
        }
        else if (idx < sizeof(line))
        {
            line[idx++] = c;
            hif_putc(c);
        }
        else
        {
            /* buffer full, throw away */
            idx = 0;
        }
    }

    return rv;
}

void process_commands(char **argv, int argc)
{
    uint8_t wbuf[32], rbuf[32], i;

    for (i = 1; i < argc; i++)
    {
        wbuf[i-1] = (uint8_t) strtol(argv[i], NULL, 16);
    }
    hif_dump(argc-1, wbuf);
    switch (argv[0][0])
    {
        case 'r':
            PRINTF("READ rlen=%d"EOL, wbuf[1]);
            i2c_master_writeread(wbuf[0], NULL, 0, rbuf, wbuf[1]);
            hif_dump(wbuf[1], rbuf);
            break;
        case 'w':
            PRINT("WRITE"EOL);
            i2c_master_writeread(wbuf[0], wbuf+1, argc-2, NULL, 0);
            break;

        case 'x':
            PRINTF("WRITE/READ wlen=%d, rlen=%d"EOL, argc-2, wbuf[argc-2]);
            i2c_master_writeread(wbuf[0], wbuf+1, argc-3, rbuf, wbuf[argc-1]);
            hif_dump(wbuf[argc-2], rbuf);
            break;

        default:
            PRINTF("Invalid command %s"EOL, argv[0]);
            break;

    }
}

/* E_O_F */
```

5.11 xmpl_isl29020.c

```
/* Copyright (c) 2009 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
```

```
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for using the I2C ISL29020 light sensor */
#include <stdlib.h>
#include "board.h"
#include "ioutil.h"
#include "i2c.h"
#include "sensors/isl29020.h"
#include "hif.h"
#include "xmpl.h"


/* === includes ============================================================ */

/* === macros ============================================================== */

/* === types =============================================================== */

/* === globals ============================================================= */
isl29020_ctx_t light1;
bool do_measure;
/* === prototypes ========================================================== */
bool process_command(char chr);

/* === functions =========================================================== */
int main(void)
{
const uint32_t br = HIF_DEFAULT_BAUDRATE;

 uint8_t cmd;
 uint8_t rv;
 uint16_t lv;
 float lv_f;

 int chr;

    mcu_init();

    /* setting up UART and adjusting the baudrate */
    hif_init(br);
    LED_INIT();
    LED_SET(0);
    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
     * been established within the host OS.
     */
    do
    {
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif
    PRINTF("\n\rISL29020 Light Sensor Example : %s : %ld bit/s\n\r", BOARD_NAME, br);

    /* init i2c bus and check presence of sensor */
    i2c_init( 4000000UL );
    rv = isl29020_init(&light1, ISL29020_ADDR);
    if (rv == 0)
    {
        PRINTF("ERROR: init, addr=0x%x\n", ISL29020_ADDR);
    }
    else
    {
        PRINTF("OK: init, addr=0x%x\n", ISL29020_ADDR);
    }
```

```
    /* 1.st configure sensor */
    cmd = 0;
    ISL29020_SET_ENABLE(cmd);
    ISL29020_SET_MODE_CONT(cmd);
    ISL29020_SET_LIGHT(cmd);
    isl29020_set_command(&light1, cmd);
    do_measure = 1;

    while (1)
    {
        chr = hif_getc();
        if (chr != -1)
        {
            do_measure = process_command(chr);
        }
        if (do_measure)
        {
            lv = isl29020_get(&light1);
            lv_f = isl29020_scale(&light1, lv);
            PRINTF("data: %-8u E[lux]: %f\n", lv, lv_f);
            WAIT_MS(1000);
            LED_TOGGLE(0);
        }
    }
}

bool process_command(char chr)
{
    int nb;
    uint8_t cmd;
    bool rv = 0;

    cmd = isl29020_get_command(&light1);
    if (chr == 'R')
    {
        PRINT("Enter range [0-3]:");
        nb = hif_get_dec_number();
        PRINTF(" %d\n", nb);

        ISL29020_SET_RANGE(cmd, nb);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 'r')
    {
        PRINT("Enter resolution [0-3]:");
        nb = hif_get_dec_number();
        PRINTF(" %d\n", nb);

        ISL29020_SET_RESOLUTION(cmd, nb);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 'l')
    {
        PRINT("detect light\n");

        ISL29020_SET_LIGHT(cmd);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 'i')
    {
        PRINT("detect infrared\n");
        ISL29020_SET_IR(cmd);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 's')
    {
        PRINT("single measurement\n");
        ISL29020_SET_MODE_SINGLE(cmd);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 'c')
    {
        PRINT("continous measurement\n");
        ISL29020_SET_MODE_CONT(cmd);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 'p')
    {
        PRINT("power down\n");
        ISL29020_SET_DISABLE(cmd);
        isl29020_set_command(&light1, cmd);
    }
    else if (chr == 'P')
    {
        PRINT("power up\n");
        ISL29020_SET_ENABLE(cmd);
        isl29020_set_command(&light1, cmd);
```

```
    }
    else if (chr == 'm' || chr == ' ')
    {
        ISL29020_SET_ENABLE(cmd);
        isl29020_set_command(&light1, cmd);
        PRINT("run measurement\n");
        rv = 1;
    }
    else if (chr == 'h')
    {
        PRINT("Help\n"\
              "R - set range 0 - 3 \n"\
              "r - set resolution 0 - 3\n"\
              "l - detect light\n"\
              "i - detect infrared\n"\
              "s - single measurement\n"\
              "c - continous measurement\n"\
              "p - power down\n"\
              "P - power up\n"\
              "m - run measurement\n"
             );
    }

    PRINTF("command: 0x%02x, "\
           "enable: %d, "\
           "mode cont: %d, "\
           "detect IR: %d, "\
           "range: %d, "\
           "resolution: %d\n",
            cmd,
            ISL29020_GET_ENABLE(cmd),
            ISL29020_GET_MODE_CONT(cmd),
            ISL29020_GET_IR(cmd),
            ISL29020_GET_RANGE(cmd),
            ISL29020_GET_RES(cmd));

    return rv;
}

/* E_O_F */
```

5.12 xmpl_key_events.c

```
/* Copyright (c) 2007 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for key event processing with a single  key */

#include "ioutil.h"
#include "timer.h"
#include "xmpl.h"

#define SHORT_PRESS_TICKS  (MSEC(100))
#define MEDIUM_PRESS_TICKS (MSEC(300))
#define LONG_PRESS_TICKS   (MSEC(800))

#define NONE_PRESS   (0)
#define SHORT_PRESS  (1)
```

```
#define MEDIUM_PRESS (2)
#define LONG_PRESS   (3)
#define CONT_PRESS   (4)

volatile uint8_t KeyEvent = 0;

static uint8_t debounce_key0(void)
{
uint8_t ret, tmp;
static uint16_t ocnt=0, ccnt=0;

    ret = NONE_PRESS;
    tmp = (KEY_GET() & 1);
    if (tmp != 0)
    {
        ocnt ++;
        if(ocnt >= LONG_PRESS_TICKS)
        {
            ocnt = LONG_PRESS_TICKS;
            ccnt++;
            if (ccnt >= MEDIUM_PRESS_TICKS)
            {
                ccnt = 0;
                ret = CONT_PRESS;
            }
        }
    }
    else
    {
        if(ocnt >= LONG_PRESS_TICKS)
        {
            ret = LONG_PRESS;
        }
        else if(ocnt >= MEDIUM_PRESS_TICKS)
        {
            ret = MEDIUM_PRESS;
        }
        else if(ocnt >= SHORT_PRESS_TICKS)
        {
            ret = SHORT_PRESS;
        }
        ccnt = 0;
        ocnt = 0;
    }

    return ret;
}

int main(void)
{
    uint8_t tmp;

    mcu_init();
    KEY_INIT();
    LED_INIT();
    TIMER_INIT();
    MCU_IRQ_ENABLE();
    while(1)
    {
        if (KeyEvent == SHORT_PRESS)
        {
            LED_TOGGLE(0);
        }

        if (KeyEvent == MEDIUM_PRESS)
        {
            LED_TOGGLE(1);
        }

        if (KeyEvent == LONG_PRESS)
        {
            LED_SET_VALUE(0);
        }
        if (KeyEvent == CONT_PRESS)
        {
            tmp = LED_GET_VALUE();
            tmp ++;
            LED_SET_VALUE(tmp);
        }
        KeyEvent = NONE_PRESS;
        SLEEP_ON_IDLE();
    }
}

ISR(TIMER_IRQ_vect)
{
    if (KeyEvent == NONE_PRESS)
```

```
    {
        KeyEvent = debounce_key0();
    }
}
/* EOF */
```

5.13 xmpl_keys.c

```
/* Copyright (c) 2007 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the KEY macros */

#include "board.h"
#include "ioutil.h"
#include "xmpl.h"

#ifdef NO_LEDS
# error "No LED Interface defined"
#endif
#ifdef NO_KEYS
# error "No KEY Interface defined"
#endif

int main(void)
{
    volatile uint8_t c;
    mcu_init();
    c = 0x55;
    LED_INIT();
    LED_SET_VALUE(c);

    KEY_INIT();
    while(1)
    {
        DELAY_MS(10);
        if( keys_debounced() )
        {
            c^=0xff;
            LED_SET_VALUE(c);
        }
    }
}
/* EOF */
```

5.14 xmpl_leds.c

```
/* Copyright (c) 2007 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
```

```
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the LED macros */

#include "board.h"
#include "ioutil.h"
#include "xmpl.h"

#ifdef NO_LEDS
# error "No LED Interface defined"
#endif

int main(void)
{
    volatile uint8_t c,x;

    mcu_init();
    LED_INIT();

    /* step 1: test the LED_TOGGLE macro */
    LED_SET_VALUE(0);
    WAIT_MS(500);
    for (c=0; c<LED_NUMBER; c++)
    {
        for (x=0; x<16; x++)
        {
            LED_TOGGLE(c);
            WAIT_MS(100);
        }
    }

    /* step 2: set each individual LED to on */
    LED_SET_VALUE(0);
    WAIT_MS(500);
    for (c=0; c<LED_NUMBER; c++)
    {
        LED_SET(c);
        WAIT_MS(500);
    }

    /* step 3: set each individual LED to off */
    for (c=0; c<LED_NUMBER; c++)
    {
        LED_CLR(c);
        WAIT_MS(500);
    }

    /* step 4: dislay BCD coded numbers from 0 to LED_MAX_VALUE */
    LED_SET_VALUE(0);
    WAIT_MS(500);
    for (c=0; c<=LED_MAX_VALUE;c++)
    {
        LED_SET_VALUE(c);
        WAIT_MS(500);
    }

    /* step 5: terminate in endless blinking loop */
    do
    {
        LED_SET_VALUE(0x55);
        WAIT_MS(500);
        LED_SET_VALUE(0xaa);
        WAIT_MS(500);
    }while(1);
}
/* EOF */
```

5.15 xmpl_lgee_acc_simple.c

5.16 xmpl_linbuf_rx.c

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the buffer functions for receive operation
 * This example expects frames to receive from xmpl_linbuf_tx.c
 */
#include <stdio.h>
#include "board.h"
#include "hif.h"
#include "radio.h"
#include "xmpl.h"

#define XMPL_FRAME_SIZE (40)

uint8_t rxbuf[XMPL_FRAME_SIZE + 2];
uint8_t buf[sizeof(buffer_t) + XMPL_FRAME_SIZE];
buffer_t *pbuf;


int main(void)
{
uint8_t frame_header[] = {0x01, 0x80, 0, 0x11,0x22,0x33,0x44};

    mcu_init();
    /* Init ressources */
    LED_INIT();
    hif_init(HIF_DEFAULT_BAUDRATE);

    radio_init(rxbuf,sizeof(rxbuf));
    MCU_IRQ_ENABLE();
    radio_set_param(RP_CHANNEL(CHANNEL));
    radio_set_state(STATE_RX);

    /* Initialize buffer structure */
    pbuf = buffer_init(buf, sizeof(buf), 0);

    while(1)
    {
        if (BUFFER_IS_LOCKED(pbuf) == true)
        {
            uint8_t c, i;
            /* do a header compare to ensure that it "our" frame */
            for(i=0; i<sizeof(frame_header);i++)
            {
                c = buffer_get_char(pbuf);
                if ( (i != 2) && (c != frame_header[i]))
                {
                    break;
                }
            }

            if (i==7)
```

```
            {
                /* if this frame belongs to us,
                   blink the LED and if available
                   send the payload to the UART */
                LED_TOGGLE(0);
                /* output at hif */
                hif_put_blk (BUFFER_PDATA(pbuf), BUFFER_SIZE(pbuf));
            }
            else
            {
                LED_TOGGLE(1);
            }
            BUFFER_RESET(pbuf,0);
            BUFFER_SET_UNLOCK(pbuf);
        }
        /* wait after this run */
        WAIT_MS(500);
    }
}


uint8_t * usr_radio_receive_frame(uint8_t len, uint8_t *frm, uint8_t lqi, int8_t ed,
       uint8_t crc)
{
    if ( BUFFER_IS_LOCKED(pbuf) == false && crc == 0)
    {
        /* copy the payload, reduced by the CRC bytes */
        buffer_append_block(pbuf, frm, len-2);
        BUFFER_SET_LOCK(pbuf);
    }
    return frm;
}

/* XEOF */
```

5.17 xmpl_linbuf_tx.c

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the buffer functions */
#include <stdio.h>
#include "board.h"
#include "hif.h"
#include "radio.h"
#include "xmpl.h"


int main(void)
{

/* buffer is smaller than a limmerick line */
#define XMPL_FRAME_SIZE (40)
/*two byte more for the CRC */
uint8_t txbuf[sizeof(buffer_t) + XMPL_FRAME_SIZE + 2];

uint8_t frame_header[] = {0x01, 0x80, 0, 0x11,0x22,0x33,0x44};
```

```
buffer_t *pbuf;
//                          1         2         3         4
//                 12345678901234567890123456789012345678901234567890123456789
char limmerick[] = "A wonderful bird is the pelican,\n\r"
                   "His bill will hold more than his belican,\n\r"
                   "He can take in his beak\n\r"
                   "Enough food for a week\n\r"
                   "But I'm damned if I see how the helican!\n\r\n\r";

char *plim;

    mcu_init();

    /* Prerequisite: Init radio */
    LED_INIT();
    radio_init(NULL, 0);
    MCU_IRQ_ENABLE();
    radio_set_param(RP_CHANNEL(CHANNEL));
    radio_set_state(STATE_TX);

    /* Initialize the buffer structure */
    pbuf = buffer_init(txbuf, sizeof(txbuf)-2, sizeof(frame_header));
    plim = limmerick;
    while(1)
    {
        /* fill buffer until '\n' or '\0' or EOF is reached */
        do
        {
            if (buffer_append_char(pbuf, *plim) == EOF)
            {
                break;
            }
            plim++;
        }
        while (*plim != 0 && *plim != '\n');

        /* finalize the buffer and transmit it */
        buffer_prepend_block(pbuf, frame_header, sizeof(frame_header));
        radio_set_state(STATE_TX);
        radio_send_frame(BUFFER_SIZE(pbuf)+ 2, BUFFER_PDATA(pbuf), 0);

        /* wait after this run */
        LED_TOGGLE(0);
        WAIT_MS(500);

        /* prepare next run */
        BUFFER_RESET(pbuf,sizeof(frame_header));
        frame_header[2]++;
        if (*plim == 0) plim = limmerick;
    }
}
/* XEOF */
```

5.18 xmpl_lm73.c

```
/* Example for using the I2C ISL29020 light sensor */
#include <stdlib.h>
#include "board.h"
#include "ioutil.h"
#include "i2c.h"
#include "sensors/lm73.h"
#include "hif.h"
#include "xmpl.h"


/* === includes ============================================================ */

/* === macros ============================================================== */

/* === types =============================================================== */

/* === globals ============================================================= */
bool do_measure;

/* === prototypes ========================================================== */
bool process_command(int chr, lm73_ctx_t *plm73);
void reg_dump(lm73_ctx_t *plm73);

/* === functions =========================================================== */

void xmpl_init(void)
{
    mcu_init();
    /* setting up UART and adjusting the baudrate */
    hif_init(HIF_DEFAULT_BAUDRATE);
    LED_INIT();
    LED_SET(0);
    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
     * been established within the host OS.
     */
    do
    {
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif

    /* init i2c bus and check presence of sensor */
    i2c_init( 4000000UL );

}

int main(void)
{
lm73_ctx_t lm73;
uint16_t temp_raw;
float temp_fl;
uint8_t status, config;
int chr;

    xmpl_init();

    PRINTF("\n\rLM73 Temerature Sensor Example : %s : %ld bit/s\n\r",
            BOARD_NAME, HIF_DEFAULT_BAUDRATE);

    sensor_create_lm73(&lm73, false, LM73_ADDR, 0);
    if (lm73.g.last_error != SENSOR_ERR_OK)
    {
        PRINTF("ERROR[%d]: init, addr=0x%x\n", lm73.g.last_error, LM73_ADDR);
    }
    else
    {
        PRINTF("OK: init, addr=0x%x\n", LM73_ADDR);
    }

    do_measure = 1;

    while (1)
    {
        chr = hif_getc();
        if (chr != -1)
        {
            do_measure = process_command(chr, &lm73);
        }
        if (do_measure)
        {
            temp_raw = lm73_get(&lm73);
            temp_fl = lm73_scale(temp_raw);
```

```
            DELAY_US(10);
            config =  lm73_byte_read(&lm73, LM73_PTR_CFG);
            DELAY_US(10);
            status =  lm73_byte_read(&lm73, LM73_PTR_STATUS);


            PRINTF("temp_raw=%d %f", temp_raw, temp_fl);
            if ( (config & LM73_CFG_PWR_DOWN) )
            {
                PRINT(" PWR_DOWN");
            }
            if ((status & LM73_STATUS_DAV) == 0)
            {
                PRINT(" NO:DATA");
            }
            if (status & LM73_STATUS_TLOW)
            {
                PRINT(" TLOW");
            }
            if (status & LM73_STATUS_THIGH)
            {
                PRINT(" THIGH");
            }
            PRINT("\n");
            WAIT_MS(1000);
            LED_TOGGLE(0);
        }
    }
}

bool process_command(int chr, lm73_ctx_t *plm73)
{
    uint8_t tmp, res;
    uint16_t t_limit;
    bool rv = 0;

    if (chr == 'P')
    {
        PRINT("power up\n");
        tmp = lm73_byte_read(plm73, LM73_PTR_CFG);
        tmp &= ~LM73_CFG_PWR_DOWN;
        DELAY_US(LM73_T_BUS_FREE_US);
        lm73_byte_write(plm73, LM73_PTR_CFG, tmp);
    }
    else if (chr == 'p')
    {
        PRINT("power down\n");
        tmp = lm73_byte_read(plm73, LM73_PTR_CFG);
        tmp |= LM73_CFG_PWR_DOWN;
        DELAY_US(LM73_T_BUS_FREE_US);
        lm73_byte_write(plm73, LM73_PTR_CFG, tmp);
    }
    else if (chr == 'o')
    {
        PRINT("one shot conversion\n");
        tmp = lm73_byte_read(plm73, LM73_PTR_CFG);
        tmp |= (LM73_CFG_PWR_DOWN | LM73_CFG_ONE_SHOT);
        DELAY_US(LM73_T_BUS_FREE_US);
        lm73_byte_write(plm73, LM73_PTR_CFG, tmp);
        rv = 1;
    }
    else if (chr == 'r')
    {
        PRINT("register dump\n");
        reg_dump(plm73);
    }
    else if (chr == 'L')
    {
        PRINT("enter low limit in °C:");
        t_limit = hif_get_dec_number();
        PRINTF(" %d°C, reg=0x%04x\n", t_limit, t_limit * LM73_TEMP_TO_REG_SCALE);
        lm73_set_lower_limit(plm73, t_limit * LM73_TEMP_TO_REG_SCALE );
    }
    else if (chr == 'H')
    {
        PRINT("enter high limit in °C:");
        t_limit = hif_get_dec_number();
        PRINTF(" %d°C, reg=0x%04x\n", t_limit, t_limit * LM73_TEMP_TO_REG_SCALE);
        lm73_set_upper_limit(plm73, t_limit * LM73_TEMP_TO_REG_SCALE );
    }
    else if (chr == 'R')
    {
        PRINT("enter resolution [0,1,2,3]:");
        res = hif_get_dec_number();
        PRINTF(" %d\n", res);
        tmp = lm73_byte_read(plm73, LM73_PTR_STATUS);
        tmp &= ~LM73_STATUS_RES;
```

```
        tmp |= ((res << 5) & LM73_STATUS_RES);
        DELAY_US(LM73_T_BUS_FREE_US);
        lm73_byte_write(plm73, LM73_PTR_STATUS, tmp);
    }
    else if (chr == 'a')
    {
        PRINT("clear alert bit\n");
        tmp = lm73_byte_read(plm73, LM73_PTR_CFG);
        tmp |= LM73_CFG_ALERT_RST;
        DELAY_US(LM73_T_BUS_FREE_US);
        lm73_byte_write(plm73, LM73_PTR_CFG, tmp);
        rv = 1;
    }
    else if (chr == 'm' || chr == ' ')
    {
        PRINT("run measurement\n");
        rv = 1;
    }
    else if (chr == 'h')
    {
        PRINT("Help\n"\
              "P/p - power up/down\n"\
              "o   - one shot conversion\n"\
              "r   - print register dump\n"\
              "H/L - set high/low limit\n"\
              "R   - set resolution\n"\
              "a   - reset alert bit\n"\
              "m   - run measurement\n"
             );
    }
    return rv;
}

void reg_dump(lm73_ctx_t *plm73)
{
    uint8_t tmp;
    uint16_t temperature;

    temperature = lm73_word_read(plm73, LM73_PTR_TEMP);
    PRINTF("\n"\
           " TEMP[%d]   = 0x%04x\n",
           LM73_PTR_TEMP, temperature, temperature * LM73_REG_TO_TEMP_SCALE );

    tmp = lm73_byte_read(plm73, LM73_PTR_CFG);
    PRINTF(" CONFIG[%d] = 0x%02x", LM73_PTR_CFG, tmp);
    if (tmp & LM73_CFG_ONE_SHOT)
    {
        PRINT(" ONE_SHOT");
    }
    if (tmp & LM73_CFG_ALERT_EN)
    {
        PRINT(" ALERT_EN");
    }
    if (tmp & LM73_CFG_ALERT_POL)
    {
        PRINT(" ALERT_POL");
    }
    if (tmp & LM73_CFG_ALERT_RST)
    {
        PRINT(" ALERT_RST");
    }
    if (tmp & LM73_CFG_PWR_DOWN)
    {
        PRINT(" PWR_DOWN");
    }
    PRINT("\n");

    temperature = lm73_word_read(plm73, LM73_PTR_THIGH);
    PRINTF(" THIGH[%d]  = 0x%04x t_high = %f°C\n",
           LM73_PTR_THIGH, temperature, temperature * LM73_REG_TO_TEMP_SCALE);

    temperature = lm73_word_read(plm73, LM73_PTR_TLOW);
    PRINTF(" TLOW[%d]   = 0x%04x t_high = %f°C\n",
           LM73_PTR_TLOW, temperature, temperature * LM73_REG_TO_TEMP_SCALE);

    tmp = lm73_byte_read(plm73, LM73_PTR_STATUS);
    PRINTF(" STATUS[%d] = 0x%02x", LM73_PTR_STATUS, tmp);
    if (tmp & LM73_STATUS_DAV)
    {
        PRINT(" DAV");
    }
    if (tmp & LM73_STATUS_TLOW)
    {
        PRINT(" TLOW");
    }
    if (tmp & LM73_STATUS_THIGH)
    {
```

```
        PRINT(" THIGH");
    }
    if (tmp & LM73_STATUS_ALERT)
    {
        PRINT(" ALERT");
    }
    PRINTF(" 1%cbit", ((tmp & LM73_STATUS_RES) >> 5) + '1');
    PRINT("\n");

    PRINTF(" ID[%d]     = 0x%04x\n\n",
           LM73_PTR_ID, lm73_word_read(plm73, LM73_PTR_ID));
}

/* E_O_F */
```

5.19 xmpl_ow.c

```
/* Copyright (c) 2015 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for use of the One-Wire functions */
#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>

#include "board.h"
#include "ioutil.h"
#include "ow.h"
#include "hif.h"
#include "xmpl.h"


/* === macros ========================================================== */
#define PROMPT() PRINT("\n\row>>> ")
#define LINE_SIZE (80)
#define MAX_ARGS  (16)

/* === globals ========================================================= */
char *argv[MAX_ARGS];
int argc;

/* === prototypes ====================================================== */
bool process_input(int c);
void process_commands(char **argv, int argc);

/* === implementation ================================================== */
int main(void)
{
 const uint32_t br = HIF_DEFAULT_BAUDRATE;
 int chr;

    mcu_init();
    /* setting up UART and adjusting the baudrate */
    hif_init(br);
    LED_INIT();
    LED_SET(0);
    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
```

```
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
     * been established within the host OS.
     */
    do
    {
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif
    ow_init();

    /* using the basic hif_xxx functions */
    PRINTF("\n\r1-Wire Example : %s : %ld bit/s\n\r", BOARD_NAME, br);

    PROMPT();
    do
    {
        chr = hif_getc();
        if (process_input(chr))
        {
            if (argc)
            {
                process_commands(argv, argc);
            }
            PROMPT();
        }
    }
    while (1);

}

bool process_input(int c)
{
    static char line[LINE_SIZE];
    static int idx;

    bool rv = false;
    if (c != EOF)
    {
        if (c == '\n' || c == '\r')
        {
            line[idx] = 0;
            idx = 0;
            hif_putc('\n');
            argc = hif_split_args(line, MAX_ARGS, argv);
            rv = true;
        }
        else if (idx < sizeof(line))
        {
            line[idx++] = c;
            hif_putc(c);
        }
        else
        {
            /* buffer full, throw away */
            idx = 0;
        }
    }

    return rv;

}

void process_commands(char **argv, int argc)
{
    uint8_t wbuf[32], rbuf[32], i;

    for (i = 1; i < argc; i++)
    {
        wbuf[i-1] = (uint8_t) strtol(argv[i], NULL, 16);
    }
    switch (argv[0][0])
    {
        case 'R':
            i = ow_reset();
            PRINTF("ow_reset: %d\n", i);
            break;

        case 'r':
            PRINTF("READ rlen=%d:", wbuf[0]);
            for(i = 0; i < wbuf[0]; i++)
            {
                PRINTF("%02x ", ow_byte_read());
            }
            break;
```

```
        case 'w':
            PRINT("WRITE: ");
            for (i = 0; i < argc-1; i++)
            {
                PRINTF("%02x ", wbuf[i]);
                ow_byte_write(wbuf[i]);
            }
            break;

        case 's':
            PRINT("SEARCH:\nnb: a7 a6 a5 a4 a3 a2 a1 a0\n");
            uint8_t first = 1;
            uint8_t dev_present;
            uint8_t devnb = 0;
            do
            {
                dev_present = ow_master_searchrom((ow_serial_t*)rbuf, first);
                first = 0;
                PRINTF("% 2d:"\
                       " %02x %02x %02x %02x"\
                       " %02x %02x %02x %02x\n",
                        devnb++,
                        rbuf[7], rbuf[6], rbuf[5], rbuf[4],
                        rbuf[3], rbuf[2], rbuf[1], rbuf[0]);
                if ( !ow_crc_valid(rbuf, 8) )
                {
                    PRINT("invalid CRC - exit search\n");
                    break;
                }
            }
            while(dev_present);
            break;

        case 'h':
            PRINT("R             : reset\n"\
                  "r <n>         : read <n> bytes\n"\
                  "w <a> <b> ... : write byte <a>, <b>, ...\n"\
                  "s             : search devices\n"\
                  "h             : show help\n"
        );

        default:
            PRINTF("invalid command: \"%s\"\n", argv[0]);
            break;

    }
    PRINT("\n");

}

/* E_O_F */
```

5.20 xmpl_radio_range.c

This application sends periodically a test frame and is the rest of the time in receive mode. All parameters of the received frames (ED, LQI, sequence number) are printed to the host interface.

The LEDs have the following meaning:

- LED0 flashes and signalizes the transmission of a frame.
- LED1 toggles with each received frame.

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
```

```
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the radio and ioutil functions for a simple range test */

#include <stdio.h>
#include "board.h"
#include "hif.h"
#include "radio.h"
#include "xmpl.h"


/* === Macros ============================================================ */
#define T_TX_PERIOD MSEC(500)
#define RT_ADDR (0x1234)
#define RT_PANID (0xcafe)
#define GET_CFG_ADDR() (0xfffc)
#define ABSDIFF(a,b) ( (a>b) ? a - b : b - a )
#define FRAME_CTRL (0x8841)
/* === Types ============================================================= */
time_t tmr_transmit(timer_arg_t t);

typedef struct
{
    uint16_t fctl;
    uint8_t  seq;
    uint16_t pan;
    uint16_t dst;
    uint16_t src;
    /* frame payload */
    uint8_t  keycnt;
    uint8_t  data[8];
    uint16_t crc;
} rt_frame_t;

/* === Globals =========================================================== */
rt_frame_t TxFrame = {
        .fctl = FRAME_CTRL,
        .pan = RT_PANID,
        .dst = RT_ADDR,
        .src = 0xffff
};
uint8_t RxFrame[MAX_FRAME_SIZE];
volatile bool do_init = true;
volatile uint8_t keycnt = 0;



/* === Implementation ==================================================== */

void app_init(void)
{
    mcu_init();
    LED_INIT();
    KEY_INIT();
    hif_init(HIF_DEFAULT_BAUDRATE);
    timer_init();
    radio_init(RxFrame, sizeof(RxFrame));
    radio_set_state(STATE_OFF);
    radio_set_param(RP_CHANNEL(CHANNEL));
    radio_set_param(RP_IDLESTATE(STATE_RXAUTO));
    radio_set_param(RP_SHORTADDR(RT_ADDR));
    radio_set_param(RP_PANID(RT_PANID));
#if RADIO_TYPE == RADIO_AT86RF212
    radio_set_param(RP_DATARATE(OQPSK100));
#endif
}


int main(void)
{
node_config_t nc = {.short_addr = 0, .pan_id = 0};

    /* setup hardware */
    app_init();

    /* copy flash settings */
```

```
    get_node_config(&nc);
    TxFrame.src = nc.short_addr;
    MCU_IRQ_ENABLE();

    PRINTF("Simple Range Test\n\r This is node: 0x%04x TX-FRAME: %d\n\r",
           nc.short_addr, sizeof(TxFrame));
    PRINTF("PWR: 0x%x CHAN: %d\r\n",
            trx_reg_read(RG_PHY_TX_PWR), trx_bit_read(SR_CHANNEL));

    do_init = true;
    timer_start(tmr_transmit,T_TX_PERIOD,0);

    while(1)
    {
        DELAY_MS(10);
        if( keys_debounced() )
        {
            LED_TOGGLE(0);
            keycnt ++;
        }

    }
}

time_t tmr_transmit(timer_arg_t t)
{
    radio_set_state(STATE_TXAUTO);
    TxFrame.keycnt = keycnt;
    radio_send_frame(sizeof(TxFrame) , (uint8_t *)&TxFrame, 0);
    LED_TOGGLE(0);
    return 0;
}

uint8_t * usr_radio_receive_frame(uint8_t len, uint8_t *frm, uint8_t lqi, int8_t ed,
       uint8_t crc)
{

rt_frame_t *pfrm;

static struct rxstatus_tag
{
    uint8_t  seq;
    uint16_t lostpckts;
    uint16_t rxdpckts;
    uint16_t src;
    uint8_t  keycnt;
    uint8_t lqi;
    int8_t ed;

} rxstatus;

bool doprint;

    LED_TOGGLE(1);

    pfrm = (rt_frame_t *) frm;
    doprint = false;

    /* verify if one of the variables has changed. */


    if (do_init == true)
    {
        do_init = false;
        rxstatus.seq = pfrm->seq;
        rxstatus.rxdpckts = 0;
        rxstatus.lostpckts = 0;

    }

    rxstatus.rxdpckts ++;

    if(pfrm->seq != rxstatus.seq)
    {
        /* improve lost packet counting with timestamp */
        rxstatus.lostpckts += ABSDIFF(pfrm->seq, rxstatus.seq);
        doprint = true;
    }

    if(pfrm->src != rxstatus.src)
    {
        rxstatus.src = pfrm->src;
        doprint = true;
    }
    if(pfrm->keycnt != rxstatus.keycnt)
    {
        rxstatus.keycnt = pfrm->keycnt;
```

```
        doprint = true;
    }
    if(lqi != rxstatus.lqi)
    {
        rxstatus.lqi = lqi;
        doprint = true;
    }
    if(ed != rxstatus.ed)
    {
        rxstatus.ed = ed;
        doprint = true;
    }

    if (doprint == true)
    {
        PRINTF(":sta 0x%04x seq %d key %d lqi %d ed %d rxd %d lost %d \n\r",
                pfrm->src, pfrm->seq, pfrm->keycnt, lqi, ed,
                rxstatus.rxdpckts, rxstatus.lostpckts );

        rxstatus.rxdpckts = 0;
        rxstatus.lostpckts = 0;
    }

    rxstatus.seq = pfrm->seq + 1;

    return frm;
}

void usr_radio_tx_done(radio_tx_done_t status)
{
    LED_TOGGLE(0);
    if (status == TX_OK)
    {
        TxFrame.seq++;
    }
    timer_start(tmr_transmit, T_TX_PERIOD, 0);
}
/* XEOF */
```

5.21 xmpl_radio_stream.c

This example shows, how the avr-libc stdio facilities can be used to do a printf to the radio transceiver. The data printed to the transceiver can be watched on a terminal either with the wuart application or with xmpl_↩ linbuf_rx.c.

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the radio stream functions */
#include <stdio.h>
#include "board.h"
#include "hif.h"
#include "radio.h"
#include "xmpl.h"

#define PROMPT() PRINTF("\n\ruracoli[%02d]> ",lc++)
```

```
void incb(buffer_t *pbuf);
void outcb(buffer_t *pbuf);

buffer_stream_t Rstream;
uint8_t frame_header[] = {0x01, 0x80, 0, 0x11,0x22,0x33,0x44};
#define XMPL_FRAME_SIZE (40)
uint8_t ibuf[sizeof(buffer_t) + XMPL_FRAME_SIZE + 2];
uint8_t obuf[sizeof(buffer_t) + XMPL_FRAME_SIZE + 2];

//volatile buffer_t *pibuf;
int main(void)
{
uint8_t rxbuf[MAX_FRAME_SIZE];
uint8_t cnt = 0;

    mcu_init();
    /* setup buffers */
    buffer_init(ibuf, sizeof(ibuf)-2, 0);
    buffer_init(obuf, sizeof(obuf)-2, sizeof(frame_header));

    /* setup buffer stream structure and stdio */
    buffer_stream_init(&Rstream, &incb, &outcb);
    /* todo add buffer assignment as parameters to buffer_stream_init! */
    Rstream.pbufin  = (buffer_t *)ibuf;
    Rstream.pbufout = (buffer_t *)obuf;
    stdout = stdin = &Rstream.bstream;

    /* setup hardware */
    LED_INIT();
    radio_init(rxbuf, MAX_FRAME_SIZE);
    radio_set_state(STATE_OFF);
    radio_set_param(RP_CHANNEL(CHANNEL));
    radio_set_param(RP_IDLESTATE(STATE_RX));
    MCU_IRQ_ENABLE();
    printf_P(PSTR("Hello World %d\n\r"),cnt++);

    while(1)
    {
        WAIT_MS(10);

        int c;
        c = getchar();
        if ( ('a' <= c) && (c <= 'z'))
        {
            printf_P(PSTR(":%c:\n\r"),c);
        }
    }
}


void outcb(buffer_t *pbuf)
{
static uint8_t frame_header[] = {0x01, 0x80, 0, 0x11,0x22,0x33,0x44};
char lastchar;
    //LED_TOGGLE(0);

    lastchar = BUFFER_LAST_CHAR(pbuf);
    if ((BUFFER_FREE_AT_END(pbuf) < 1) || (lastchar == '\r'))
    {
        /* prepare send */
        buffer_prepend_block(pbuf, frame_header, sizeof(frame_header));
        radio_set_state(STATE_TX);
        radio_send_frame(BUFFER_SIZE(pbuf)+ 2, BUFFER_PDATA(pbuf), 0);
        /* clean buffer */
        BUFFER_RESET(pbuf, sizeof(frame_header));
        frame_header[2]++;
        /* blink LED if done. */
        //LED_TOGGLE(0);
    }
}


void incb(buffer_t *pbuf)
{
uint8_t sz;
    sz = BUFFER_SIZE(pbuf);
    if (sz < 1)
    {
        /* buffer is now empty, free it. */
MCU_IRQ_DISABLE();
        BUFFER_RESET(pbuf,0);
        BUFFER_SET_UNLOCK(pbuf);
MCU_IRQ_ENABLE();
    }
}
```

```
uint8_t * usr_radio_receive_frame(uint8_t len, uint8_t *frm, uint8_t lqi, int8_t ed,
       uint8_t crc)
{
uint16_t fctl;
uint8_t hlength;
    LED_TOGGLE(1);


    if ( BUFFER_IS_LOCKED(Rstream.pbufin) == false && crc == 0)
    {
        fctl = *(uint16_t*)frm;
        /* copy the payload, reduced by the CRC bytes */
        buffer_append_block(Rstream.pbufin, frm, len-2);
        hlength = 3;
        hlength += ((fctl & FCTL_DST_MASK) == FCTL_DST_LONG) ? 10:0;
        hlength += ((fctl & FCTL_DST_MASK) == FCTL_DST_SHORT) ? 4:0;
        hlength += ((fctl & FCTL_SRC_MASK) == FCTL_SRC_LONG) ? 10:0;
        hlength += ((fctl & FCTL_SRC_MASK) == FCTL_SRC_SHORT) ? 4:0;
        if (fctl & FCTL_IPAN_MASK)
        {
            hlength -= (fctl & FCTL_SRC_MASK) ? 2:0;
        }
        BUFFER_ADVANCE(Rstream.pbufin,hlength-1);
        printf("rx=%d\n\r", BUFFER_SIZE(Rstream.pbufin));
        BUFFER_SET_LOCK(Rstream.pbufin);
    }
    return frm;
}
/* XEOF */
```

5.22 xmpl_rtc.c

```
/* Copyright (c) 2014 Axel Wachtler, Daniel Thiele
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for using the rtc module */
#include "board.h"
#include "rtc.h"
#include "ioutil.h"

void rtc_tick(void)
{
    LED_SET(0);
    DELAY_MS(5);
    LED_CLR(0);
}

int main(void)
{

    mcu_init();
    LED_INIT();
    rtc_init(rtc_tick);
    MCU_IRQ_ENABLE();
    rtc_start();
    for(;;) {
```

```
        SLEEP_ON_IDLE();
    }
}

/*EOF*/
```

5.23 xmpl_sensor.c

```
/* Copyright (c) 2013 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example use of the sensor functions */

#include "board.h"
#include "sensor.h"
#include "ioutil.h"
#include "xmpl.h"

int main(void)
{
    uint8_t buf[128], rv;
    char line[128];
    uint8_t data_size, one_shot = 0, i;


    /* init ressources */
    mcu_init();
    hif_init(HIF_DEFAULT_BAUDRATE);
    LED_INIT();
    LED_SET(0);
    MCU_IRQ_ENABLE();
    i2c_init( 4000000UL );

    /* init all sesnors, individual sensors are adressed with a number
     * from 0 to NB_OF_SENSORS - 1.
     */
    rv = create_board_sensors();

    PRINTF("sensor_init: sensors=%d\n", rv );

    PRINTF("number of sensors: %d\n", sensor_get_number());
    for (i = 0; i < sensor_get_number(); i++)
    {
        PRINTF("  %d : id=%d name=%s type=%s data_sz=%d\n",
               i, sensor_get_id(i),
               sensor_get_name(i),
               sensor_get_type(i),
               sensor_get(i, NULL));
    }
    /* probe for the max. size of the data record with a NULL pointer */
    data_size = sensor_get(-1, NULL);
    PRINTF("total data size: %d\n", data_size);

    /* for special configurations of individual sensors, you have to call the
     * low level functions from the sensor directly.
     */
    while (1)
    {
```

```
        sensor_trigger(-1, one_shot);
        DELAY_MS(10);
        sensor_get(-1, buf);
        if (one_shot)
        {
            sensor_sleep(-1);
        }
        PRINTF("decode: %s\n",sensor_decode(buf, line, sizeof(line)));

        DELAY_MS(2000);
    }
}
```

5.24 xmpl_timer.c

```
/* Copyright (c) 2007 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for using the timer macros */
#include "board.h"
#include "timer.h"
#include "ioutil.h"

#ifndef NO_TIMER
int main(void)
{
    mcu_init();
    LED_INIT();
    TIMER_INIT();
    MCU_IRQ_ENABLE();
    while(1)
    {
        DELAY_MS(100);
    }
}

ISR(TIMER_IRQ_vect)
{
static time_t  ticktime = 0;
    ticktime ++;
    /* LED_0 blinks with the calling frequency of the IRQ routine */
    LED_TOGGLE(0);
    if (ticktime > MSEC(500))
    {
        /* LED_1 blinks with a period of 1s */
        LED_TOGGLE(1);
        ticktime  = 0;
    }
}
#endif
/*EOF*/
```

5.25 xmpl_timer_callback.c

```
/* Copyright (c) 2007 Axel Wachtler
```

```
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for using the timer macros */
#include "board.h"
#include "timer.h"
#include "hif.h"
#include "ioutil.h"

/* === prototypes ============================================================ */
void xmpl_init(void);
time_t usr_timer_cb(timer_arg_t ta);



int main(void)
{
    uint8_t ta;

    /* init peripherals */
    xmpl_init();

    ta = 42;
    timer_start( usr_timer_cb, MSEC(1000), (timer_arg_t) ta);

    while(1)
    {
        /*todo: add sleep macros here */
    }
}

void xmpl_init(void)
{
    mcu_init();
    LED_INIT();
    hif_init(HIF_DEFAULT_BAUDRATE);
    timer_init();
    MCU_IRQ_ENABLE();
}

time_t usr_timer_cb(timer_arg_t ta)
{
    uint8_t tvar;
    tvar = (uint8_t) ta;
    PRINTF("usr_timer_cb: tvar=%d now = %ld ticks\n", tvar, timer_systime());
    LED_TOGGLE(1);
    /* restart timer afer 1000ms */
    return MSEC(1000);
}

/*EOF*/
```

5.26 xmpl_trx_base.c

This example illustrates the use of the basic radio transceiver access functions. It can also be used to test a new board definition.

```
/* Copyright (c) 2008 Axel Wachtler
```

```
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for accessing the transceiver */

#include "board.h"
#include "transceiver.h"
#include "ioutil.h"
#include "xmpl.h"

static volatile trx_regval_t irq_cause;

int main(void)
{
static volatile trx_regval_t rval;

    mcu_init();
    /* This will stop the application before initializing the radio transceiver
     * (ISP issue with MISO pin, see FAQ)
     */
    trap_if_key_pressed();

    /* init peripherals */
    LED_INIT();
    trx_io_init(SPI_RATE_1_2);
    LED_SET_VALUE(LED_MAX_VALUE);
    LED_SET_VALUE(0);

    /* Step 1: reset transceiver */
    DELAY_US(TRX_INIT_TIME_US);
    TRX_RESET_LOW();
    TRX_SLPTR_LOW();
    DELAY_US(TRX_RESET_TIME_US);
    TRX_RESET_HIGH();
    LED_SET_VALUE(1);
    DELAY_MS(10);

    /* Step 2: check if correct transceiver is present
     * verify if the part and version ID's are correct.
     */
    rval = trx_reg_read(RG_PART_NUM);
    ERR_CHECK(RADIO_PART_NUM != rval);

    rval = trx_reg_read(RG_VERSION_NUM);
    ERR_CHECK_DIAG(RADIO_VERSION_NUM !=rval,2);


    /* Step 3: try a write command on an arbitrary RW register.
     */
    trx_reg_write(RG_IEEE_ADDR_0,42);
    rval = trx_reg_read(RG_IEEE_ADDR_0);
    ERR_CHECK_DIAG((42!=rval),3);


    /* Step 4: go to state TRX_OFF
     * bring the transceiver in state TRX_OFF and verify it.
     */
    trx_reg_write(RG_TRX_STATE, CMD_TRX_OFF);
    DELAY_US(TRX_INIT_TIME_US);
    DELAY_MS(1000);
    rval = trx_bit_read(SR_TRX_STATUS);
    ERR_CHECK_DIAG((TRX_OFF!=rval),4);
```

```
    /* Step 5: check SRAM access */

    /* Step 6: check IRQ
     * enable PLL-LOCK IRQ and transceiver IRQ at MCU
     * switch to PLL_ON state (which causes lock IRQ)
     * wait until the interrupt happened
     */
    trx_reg_write(RG_IRQ_MASK, TRX_IRQ_PLL_LOCK);
    irq_cause = 0;
    MCU_IRQ_ENABLE();

    trx_reg_write(RG_TRX_STATE,CMD_PLL_ON);

    DELAY_US(TRX_PLL_LOCK_TIME_US);
    DELAY_MS(200);
    ERR_CHECK_DIAG((0==irq_cause),6);
    /* done */
    LED_SET_VALUE(0);
    while(1)
    {
        WAIT500MS();
        LED_TOGGLE(0);
    }
}

#if defined(TRX_IF_RFA1)
ISR(TRX24_PLL_LOCK_vect)
{
    irq_cause |= TRX_IRQ_PLL_LOCK;
}
#else
ISR(TRX_IRQ_vect)
{
    /* reading IRQ_STATUS acknowledges the interrupt */
    irq_cause |= trx_reg_read(RG_IRQ_STATUS);
    LED_SET(1);
}
#endif

/* EOF */
```

5.27 xmpl_trx_echo.c

This example shows, how frames are received and retransmitted if their CRC was correct. This example is esentially a combination from pgXmplTrxTx and pgXmplTrxRx. The retransmission of the frames is initiated within the transceiver interrupt routine without any CCA.

Note

Use this example with **care** !!! You can mess up your sensor network easily.

```
/* Copyright (c) 2009 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
```

```
/* Example for echoing received frames */

#include "board.h"
#include "transceiver.h"
#include "ioutil.h"
#include  <util/crc16.h>
#include "xmpl.h"


static uint8_t rxfrm[MAX_FRAME_SIZE];
static volatile uint8_t rxcnt;
static uint8_t state;

int main(void)
{
trx_regval_t rval;

    mcu_init();

    /* This will stop the application before initializing the radio transceiver
     * (ISP issue with MISO pin, see FAQ)
     */
    trap_if_key_pressed();

    /* Step 0: init MCU peripherals */
    LED_INIT();
    trx_io_init(SPI_RATE_1_2);
    LED_SET_VALUE(LED_MAX_VALUE);
    LED_SET_VALUE(0);

    /* Step 1: initialize the transceiver */
    TRX_RESET_LOW();
    TRX_SLPTR_LOW();
    DELAY_US(TRX_RESET_TIME_US);
    TRX_RESET_HIGH();
    trx_reg_write(RG_TRX_STATE,CMD_TRX_OFF);
    DELAY_US(TRX_INIT_TIME_US);
    rval = trx_bit_read(SR_TRX_STATUS);
    ERR_CHECK(TRX_OFF!=rval);
    LED_SET_VALUE(1);

    /* Step 2: setup transmitter
     * - configure radio channel
     * - go into RX state,
     * - enable "receive end" IRQ
     */
    trx_bit_write(SR_CHANNEL,CHANNEL);
    trx_reg_write(RG_TRX_STATE,CMD_RX_ON);
#if defined(TRX_IRQ_TRX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TRX_END);
#elif defined(TRX_IRQ_RX_END) && defined(TRX_IRQ_TX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_RX_END | TRX_IRQ_TX_END);
#else
#  error "Unknown IRQ bits"
#endif
    MCU_IRQ_ENABLE();
    LED_SET_VALUE(2);

    /* Step 3: Going to receive frames */
    rxcnt = 0;

    LED_SET_VALUE(0);
    while(1);
}

#if defined(TRX_IF_RFA1)
ISR(TRX24_RX_END_vect)
{
uint8_t *pfrm, tmp, flen;
uint16_t crc;

    LED_SET_VALUE(0);
    /* upload frame and check for CRC16 validity */
    pfrm = rxfrm;
    flen = trx_frame_read(pfrm, sizeof(rxfrm), NULL);
    tmp = flen;
    crc = 0;
    do
    {
        crc = _crc_ccitt_update(crc, *pfrm++);
    }
    while(tmp--);
    /* if crc is correct, update RX frame counter */
    if (crc == 0)
    {
        rxcnt ++;
        /* echo the frame if CRC is valid*/
```

```
        trx_reg_write(RG_TRX_STATE,CMD_FORCE_TRX_OFF);
        trx_reg_write(RG_TRX_STATE,CMD_PLL_ON);
        /*invert sequence number, just to show if it is from pinger */
        rxfrm[2] ^=0xff;
        DELAY_US(16); /* wait 1 symbol, XXX check this timing */
        TRX_SLPTR_HIGH();
        TRX_SLPTR_LOW();
        trx_frame_write(flen,rxfrm);
        state = STATE_TX;
    }

    /* display current frame count */
    LED_SET_VALUE(rxcnt);

}

ISR(TRX24_TX_END_vect)
{

    trx_reg_write(RG_TRX_STATE,CMD_RX_ON);
    state = STATE_RX;
}
#else  /* !RFA1 */
ISR(TRX_IRQ_vect)
{
static volatile trx_regval_t irq_cause;
uint8_t *pfrm, tmp, flen;
uint16_t crc;

    irq_cause = trx_reg_read(RG_IRQ_STATUS);
    if (irq_cause & TRX_IRQ_TRX_END)
    {

        if (state == STATE_TX)
        {
            trx_reg_write(RG_TRX_STATE,CMD_RX_ON);
            state = STATE_RX;
        }
        else
        {
            LED_SET_VALUE(0);
            /* upload frame and check for CRC16 validity */
            pfrm = rxfrm;
            flen = trx_frame_read(pfrm, sizeof(rxfrm), NULL);
            tmp = flen;
            crc = 0;
            do
            {
                crc = _crc_ccitt_update(crc, *pfrm++);
            }
            while(tmp--);
            /* if crc is correct, update RX frame counter */
            if (crc == 0)
            {
                rxcnt ++;
                /* echo the frame if CRC is valid*/
                trx_reg_write(RG_TRX_STATE,CMD_FORCE_TRX_OFF);
                trx_reg_write(RG_TRX_STATE,CMD_PLL_ON);
                /*invert sequence number, just to show if it is from pinger */
                rxfrm[2] ^=0xff;
                DELAY_US(16); /* wait 1 symbol, XXX check this timing */
                TRX_SLPTR_HIGH();
                TRX_SLPTR_LOW();
                trx_frame_write(flen,rxfrm);
                state = STATE_TX;
            }

            /* display current frame count */
            LED_SET_VALUE(rxcnt);

        }
    }
}
#endif  /* RFA1 */

/* EOF */
```

5.28 xmpl_trx_rx.c

This example shows, how frames are received and their CRCR16 is checked in software.

```
/* Copyright (c) 2007 Axel Wachtler
   All rights reserved.
```

```
   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for receiving frames */

#include "board.h"
#include "transceiver.h"
#include "ioutil.h"
#include  <util/crc16.h>
#include "xmpl.h"


static uint8_t rxfrm[MAX_FRAME_SIZE];
static volatile uint8_t rxcnt;
int main(void)
{
trx_regval_t rval;

    mcu_init();

    /* This will stop the application before initializing the radio transceiver
     * (ISP issue with MISO pin, see FAQ)
     */
    trap_if_key_pressed();

    /* Step 0: init MCU peripherals */
    LED_INIT();
    trx_io_init(SPI_RATE_1_2);
    LED_SET_VALUE(LED_MAX_VALUE);
    LED_SET_VALUE(0);

    /* Step 1: initialize the transceiver */
    TRX_RESET_LOW();
    TRX_SLPTR_LOW();
    DELAY_US(TRX_RESET_TIME_US);
    TRX_RESET_HIGH();
    trx_reg_write(RG_TRX_STATE,CMD_TRX_OFF);
    DELAY_US(TRX_INIT_TIME_US);
    rval = trx_bit_read(SR_TRX_STATUS);
    ERR_CHECK(TRX_OFF!=rval);
    LED_SET_VALUE(1);

    /* Step 2: setup transmitter
     * - configure radio channel
     * - go into RX state,
     * - enable "receive end" IRQ
     */
    trx_bit_write(SR_CHANNEL,CHANNEL);
    trx_reg_write(RG_TRX_STATE,CMD_RX_ON);
#if defined(TRX_IRQ_TRX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TRX_END);
#elif defined(TRX_IRQ_RX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_RX_END);
#else
#  error "Unknown IRQ bits"
#endif
    MCU_IRQ_ENABLE();
    LED_SET_VALUE(2);

    /* Step 3: Going to receive frames */
    rxcnt = 0;

    LED_SET_VALUE(0);
```

```
    while(1);
}

#if defined(TRX_IF_RFA1)
ISR(TRX24_RX_END_vect)
{
uint8_t flen, *pfrm;
uint16_t crc;

    /* upload frame and check for CRC16 validity */
    pfrm = rxfrm;
    flen = trx_frame_read(pfrm, sizeof(rxfrm), NULL);
    crc = 0;
    do
    {
        crc = _crc_ccitt_update(crc, *pfrm++);
    }
    while(flen--);
    /* if crc is correct, update RX frame counter */
    if (crc == 0)
    {
        rxcnt ++;
    }
    /* display current rx statistics
     * LED[0] toggles with every received frame
     * LED[1:n] display the count of frames received with valid CRC
     */
    LED_SET_VALUE((rxcnt<<1) | ((LED_GET_VALUE()&1)^1));
}
#else  /* !RFA1 */
ISR(TRX_IRQ_vect)
{
static volatile trx_regval_t irq_cause;
uint8_t flen, *pfrm, tmp;
uint16_t crc;

    irq_cause = trx_reg_read(RG_IRQ_STATUS);
    if (irq_cause & TRX_IRQ_TRX_END)
    {
        /* upload frame and check for CRC16 validity */
        pfrm = rxfrm;
        flen = trx_frame_read(pfrm, sizeof(rxfrm), NULL);
        crc = 0;
        do
        {
            crc = _crc_ccitt_update(crc, *pfrm++);
        }
        while(flen--);
        /* if crc is correct, update RX frame counter */
        if (crc == 0)
        {
            rxcnt ++;
        }
        /* display current rx statistics
         * LED[0] toggles with every received frame
         * LED[1:n] display the count of frames received with valid CRC
         */
        tmp = (rxcnt<<1) | ((LED_GET_VALUE()&1)^1);
        LED_SET_VALUE(tmp);
    }
}
#endif  /* RFA1 */

/* EOF */
```

5.29 xmpl_trx_rxaack.c

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
```

```
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for receiving frames in rx_aack mode */

#include "board.h"
#include "transceiver.h"
#include "ioutil.h"
#include "xmpl.h"

static volatile bool tx_in_progress;
static volatile uint8_t tx_cnt;
static volatile uint8_t tx_fail;

#define SEQ_OFFSET     (2)
#define TX_FAIL_OFFSET (7)
#define TX_SRAM_OFFSET (1)

int main(void)
{
trx_regval_t rval;

    mcu_init();

    /* This will stop the application before initializing the radio transceiver
     * (ISP issue with MISO pin, see FAQ)
     */
    trap_if_key_pressed();

    /* Step 0: init MCU peripherals */
    LED_INIT();
    trx_io_init(SPI_RATE_1_2);
    LED_SET_VALUE(LED_MAX_VALUE);
    LED_SET_VALUE(0);

    /* Step 1: initialize the transceiver */
    TRX_RESET_LOW();
    TRX_SLPTR_LOW();
    DELAY_US(TRX_RESET_TIME_US);
    TRX_RESET_HIGH();
    trx_reg_write(RG_TRX_STATE,CMD_TRX_OFF);
    DELAY_MS(TRX_INIT_TIME_US);
    rval = trx_bit_read(SR_TRX_STATUS);
    ERR_CHECK(TRX_OFF!=rval);
    LED_SET_VALUE(1);

    /* Step 2: setup transmitter
     * - configure radio channel
     * - enable transmitters automatic crc16 generation
     * - go into RX AACK state,
     * - configure address filter
     * - enable "receive end" IRQ
     */
    trx_bit_write(SR_CHANNEL,CHANNEL);
    trx_bit_write(SR_TX_AUTO_CRC_ON,1);

    trx_reg_write(RG_PAN_ID_0,(PANID&0xff));
    trx_reg_write(RG_PAN_ID_1,(PANID>>8));

    trx_reg_write(RG_SHORT_ADDR_0,(SHORT_ADDR&0xff));
    trx_reg_write(RG_SHORT_ADDR_1,(SHORT_ADDR>>8));

    trx_reg_write(RG_TRX_STATE,CMD_RX_AACK_ON);
#if defined(TRX_IRQ_TRX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TRX_END);
#elif defined(TRX_IRQ_RX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_RX_END);
#else
#  error "Unknown IRQ bits"
#endif
    MCU_IRQ_ENABLE();
    LED_SET_VALUE(2);

    /* Step 3: send a frame each 500ms */
    tx_cnt = 0;
    tx_in_progress = false;
    LED_SET_VALUE(0);
```

```
    while(1);

}

#if defined(TRX_IF_RFA1)
ISR(TRX24_RX_END_vect)
{
static volatile trx_regval_t trac_status;

    trac_status = trx_bit_read(SR_TRAC_STATUS);

    LED_TOGGLE(0);
    if (trac_status != TRAC_SUCCESS)
    {
        LED_TOGGLE(1);
    }
}
#else  /* !RFA1 */
ISR(TRX_IRQ_vect)
{
static volatile trx_regval_t irq_cause;
static volatile trx_regval_t trac_status;

    irq_cause = trx_reg_read(RG_IRQ_STATUS);
    trac_status = trx_bit_read(SR_TRAC_STATUS);

    if (irq_cause & TRX_IRQ_TRX_END)
    {
        LED_TOGGLE(0);
        if (trac_status != TRAC_SUCCESS)
        {
            LED_TOGGLE(1);
        }
    }
}
#endif  /* RFA1 */

/* EOF */
```

5.30 xmpl_trx_tx.c

This example shows, how a frame is transmitted.

```
/* Copyright (c) 2007 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for transmitting frames */

#include "board.h"
#include "transceiver.h"
#include "ioutil.h"
#include "xmpl.h"


static volatile bool tx_in_progress;
static volatile uint8_t tx_cnt;
```

```
int main(void)
{
trx_regval_t rval;
uint8_t txfrm[] = {1,0, /* faked ieee 802.15.4 data frame control field
                         * this is just needed, that a sniffer has to display something.*/
                    42,  /* sequence counter, updated by software */
                   'h','e','l','l','o',' ','u','r','a','c','o','l','i','!', /* data */
                   'X','X' /* these bytes are overwritten from transceivers CRC generator just before sent.
      */
                   };
    mcu_init();

    /* This will stop the application before initializing the radio transceiver
     * (ISP issue with MISO pin, see FAQ)
     */
    trap_if_key_pressed();

    /* Step 0: init MCU peripherals */
    LED_INIT();
    trx_io_init(SPI_RATE_1_2);
    LED_SET_VALUE(LED_MAX_VALUE);
    LED_SET_VALUE(0);

    /* Step 1: initialize the transceiver */
    DELAY_US(TRX_INIT_TIME_US);
    TRX_RESET_LOW();
    TRX_SLPTR_LOW();
    DELAY_US(TRX_RESET_TIME_US);
    TRX_RESET_HIGH();
    trx_reg_write(RG_TRX_STATE,CMD_TRX_OFF);
    DELAY_US(TRX_INIT_TIME_US);
    rval = trx_bit_read(SR_TRX_STATUS);
    ERR_CHECK_DIAG(TRX_OFF!=rval,1);
    LED_SET_VALUE(1);

    /* Step 2: setup transmitter
     * - configure radio channel
     * - enable transmitters automatic crc16 generation
     * - go into TX state,
     * - enable "transmit end" IRQ
     */
    trx_bit_write(SR_CHANNEL,CHANNEL);
    trx_bit_write(SR_TX_AUTO_CRC_ON,1);
    trx_reg_write(RG_TRX_STATE,CMD_PLL_ON);
#if defined(TRX_IRQ_TRX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TRX_END);
#elif defined(TRX_IRQ_TX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TX_END);
#else
#  error "Unknown IRQ bits"
#endif
    MCU_IRQ_ENABLE();
    LED_SET_VALUE(2);

    /* Step 3: send a frame each 500ms */
    tx_cnt = 0;
    tx_in_progress = false;
    LED_SET_VALUE(0);

    while(1)
    {
        WAIT500MS();
        if (tx_in_progress == false)
        {
            txfrm[2] = tx_cnt;
            trx_frame_write (sizeof(txfrm), txfrm);
            tx_in_progress = true;
            TRX_SLPTR_HIGH();
            TRX_SLPTR_LOW();
            LED_SET(1);
            LED_TOGGLE(0);
        }
    }
}

#if defined(TRX_IF_RFA1)
ISR(TRX24_TX_END_vect)
{
    /* transmission completed */
    tx_in_progress = false;
    tx_cnt ++;
    LED_CLR(1);
}
#else  /* !RFA1 */
ISR(TRX_IRQ_vect)
{
static volatile trx_regval_t irq_cause;
```

```
    irq_cause = trx_reg_read(RG_IRQ_STATUS);
    if (irq_cause & TRX_IRQ_TRX_END)
    {
        /* transmission completed */
        tx_in_progress = false;
        tx_cnt ++;
        LED_CLR(1);
    }
}
#endif

/* EOF */
```

5.31 xmpl_trx_txaret.c

The example shows, how a frame is transmitted in TX_ARET mode. Within this mode, CCA is done automatically. Additionally the example illustrates, how the fram data can be directly manipulated in the SRAM (see IRQ service routine).

```
/* Copyright (c) 2008 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for transmitting frames in tx_aret mode */

#include "board.h"
#include "transceiver.h"
#include "ioutil.h"
#include "xmpl.h"

static volatile bool tx_in_progress;
static volatile uint8_t tx_cnt, fail_cnt;

#define SEQ_OFFSET     (2)
#define TX_FAIL_OFFSET (7)
#define TX_SRAM_OFFSET (1)

int main(void)
{
trx_regval_t rval;
uint8_t txfrm[] = {0x21,0x08,               /* IEEE 802.15.4 FCF:
                                               data frame with ack request */
                   42,                      /* sequence counter */
                   (PANID & 0xff), (PANID >> 8), /* destination PAN_ID */
                   (SHORT_ADDR & 0xff), (SHORT_ADDR >> 8), /* dest. short address */
                   42,                      /* TX fail counter */
                   'h','e','l','l','o',' ','u','r','a','c','o','l','i','!', /* data */
                   'X','X'    /* these bytes are overwritten from transceivers
                               * CRC generator directly before sent. */
                  };

    mcu_init();

    /* This will stop the application before initializing the radio transceiver
     * (ISP issue with MISO pin, see FAQ)
     */
    trap_if_key_pressed();
```

```
    /* Step 0: init MCU peripherals */
    LED_INIT();
    trx_io_init(SPI_RATE_1_2);
    LED_SET_VALUE(LED_MAX_VALUE);
    LED_SET_VALUE(0);

    /* Step 1: initialize the transceiver */
    TRX_RESET_LOW();
    TRX_SLPTR_LOW();
    DELAY_US(TRX_RESET_TIME_US);
    TRX_RESET_HIGH();
    trx_reg_write(RG_TRX_STATE,CMD_TRX_OFF);
    DELAY_MS(TRX_INIT_TIME_US);
    rval = trx_bit_read(SR_TRX_STATUS);
    ERR_CHECK(TRX_OFF!=rval);
    LED_SET_VALUE(1);

    /* Step 2: setup transmitter
     * - configure radio channel
     * - enable transmitters automatic crc16 generation
     * - go into TX state,
     * - enable "transmit end" IRQ
     */
    trx_bit_write(SR_CHANNEL,CHANNEL);
    trx_bit_write(SR_TX_AUTO_CRC_ON,1);
    trx_reg_write(RG_TRX_STATE,CMD_TX_ARET_ON);
#if defined(TRX_IRQ_TRX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TRX_END);
#elif defined(TRX_IRQ_TX_END)
    trx_reg_write(RG_IRQ_MASK,TRX_IRQ_TX_END);
#else
#  error "Unknown IRQ bits"
#endif
    MCU_IRQ_ENABLE();
    LED_SET_VALUE(2);

    /* Step 3: send a frame each 500ms */
    tx_cnt = 0;
    tx_in_progress = false;
    LED_SET_VALUE(0);

    while(1)
    {
        WAIT500MS();
        if (tx_in_progress == false)
        {
            txfrm[SEQ_OFFSET] = tx_cnt;
            txfrm[TX_FAIL_OFFSET] = fail_cnt;
            trx_frame_write (sizeof(txfrm), txfrm);
            tx_in_progress = true;
            TRX_SLPTR_HIGH();
            TRX_SLPTR_LOW();
            LED_SET(1);
            LED_TOGGLE(0);
        }
    }
}

#if defined(TRX_IF_RFA1)
ISR(TRX24_TX_END_vect)
{
static volatile trx_regval_t trac_status;

    trac_status = trx_bit_read(SR_TRAC_STATUS);

    tx_in_progress = false;
    if (trac_status != TRAC_SUCCESS)
    {
        fail_cnt++;
    }
    else
    {
        tx_cnt ++;
        LED_CLR(1);
    }
}
#else  /* !RFA1 */
ISR(TRX_IRQ_vect)
{
static volatile trx_regval_t irq_cause;
static volatile trx_regval_t trac_status;

    irq_cause = trx_reg_read(RG_IRQ_STATUS);
    trac_status = trx_bit_read(SR_TRAC_STATUS);

    if (irq_cause & TRX_IRQ_TRX_END)
```

```
    {
        tx_in_progress = false;
        if (trac_status != TRAC_SUCCESS)
        {
            fail_cnt++;
        }
        else
        {
            tx_cnt ++;
            LED_CLR(1);
        }
    }
}
#endif  /* RFA1 */

/* EOF */
```

5.32 xmpl_tsl2550.c

```
/* Copyright (c) 2009 Axel Wachtler
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   * Neither the name of the authors nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
   POSSIBILITY OF SUCH DAMAGE. */

/* $Id$ */
/* Example for using the I2C ISL29020 light sensor */
#include <stdlib.h>
#include "board.h"
#include "ioutil.h"
#include "i2c.h"
#include "sensors/tsl2550.h"
#include "hif.h"
#include "xmpl.h"


/* === includes ============================================================ */

/* === macros ============================================================== */

/* === types =============================================================== */

/* === globals ============================================================= */
bool do_measure;

/* === prototypes ========================================================== */
bool process_command(int chr);

/* === functions =========================================================== */

void xmpl_init(void)
{
    /* setting up UART and adjusting the baudrate */
    mcu_init();
    hif_init(HIF_DEFAULT_BAUDRATE);
    LED_INIT();
    LED_SET(0);
    MCU_IRQ_ENABLE();
#if HIF_TYPE == HIF_AT90USB
    /*
     * Wait for terminal user pressing a key so there is time to
     * attach a terminal emulator after the virtual serial port has
```

```
     * been established within the host OS.
     */
    do
    {
        inchar = hif_getc();
    }
    while (EOF == inchar);
#endif

    /* init i2c bus and check presence of sensor */
    i2c_init( 4000000UL );

}

int main(void)
{
uint8_t rv, adc0, adc1;
uint16_t lv_f;

int chr;

    xmpl_init();

    PRINTF("\n\rTSL2550 Light Sensor Example : %s : %ld bit/s\n\r",
            BOARD_NAME, HIF_DEFAULT_BAUDRATE);

    rv = tsl2550_init();
    if (rv == 0)
    {
        PRINTF("ERROR: init, addr=0x%x\n", TSL2550_ADDR);
    }
    else
    {
        PRINTF("OK: init, addr=0x%x\n", TSL2550_ADDR);
    }


    do_measure = 1;

    /* wakeup sensor */
    tsl2550_set_command(TSL2550_RD_CMD);


    while (1)
    {
        chr = hif_getc();
        if (chr != -1)
        {
            do_measure = process_command(chr);
        }
        if (do_measure)
        {
            adc0 = tsl2550_get(0);
             /* a time gap between two reads is needed, todo: check datasheet */
            DELAY_US(10);
            adc1 = tsl2550_get(1);

            if ( TSL2550_ADC_VALID(adc0) && TSL2550_ADC_VALID(adc1))
            {
                lv_f = tsl2550_scale( adc0 , adc1);
                PRINTF("adc0: 0x%02x adc1: 0x%02x E[lux]: %d\n", adc0, adc1, lv_f);
            }
            else
            {
                PRINTF("err adc0: 0x%02x adc1: 0x%02x\n", adc0, adc1);
            }
            /* conversion cycle of TSL2550 is 800ms for both adc's. */
            WAIT_MS(1000);
            LED_TOGGLE(0);
        }
    }
}

bool process_command(int chr)
{
    bool rv = 0;

    if (chr == 'p')
    {
        PRINT("power down\n");
        tsl2550_set_command(TSL2550_PWR_DOWN);
    }
    else if (chr == 'P')
    {
        PRINT("power up\n");
        tsl2550_set_command(TSL2550_RD_CMD);
    }
```

```
    else if (chr == 'e')
    {
        PRINT("use ext. range\n");
        tsl2550_set_command(TSL2550_EXT_RANGE);
    }
    else if (chr == 's')
    {
        PRINT("use std. range\n");
        tsl2550_set_command(TSL2550_STD_RANGE);
    }
    else if (chr == 'm' || chr == ' ')
    {
        PRINT("run measurement\n");
        rv = 1;
    }
    else if (chr == 'h')
    {
        PRINT("Help\n"\
              "p - power down\n"\
              "P - power up\n"\
              "e - use ext. range\n"\
              "s - use std. range\n"\
              "m - run measurement\n"
             );
    }

    return rv;
}

/* E_O_F */
```

Index