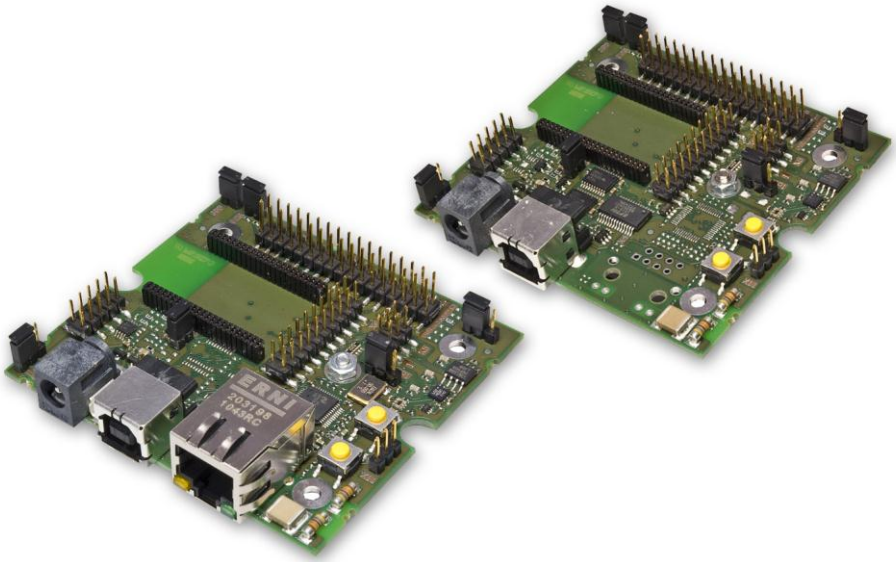
dresden elektronik

User Manual
deRFnode / deRFgateway



Document Version V1.3
2014-04-10

Table of contents

1. Overview 6

2. Application 6

3. Features 6
   3.1. Block diagram 7
   3.2. Hardware selection table 7
   3.3. Feature list 8

4. Hardware selection examples 9
   4.1. ZigBee sensor network with battery powered nodes 9
   4.2. 6LoWPAN tree-network application 9
   4.3. Point-to-Point connection for simple applications 9

5. Technical data 10
   5.1. Mechanical 10
   5.2. Operation conditions 10
   5.3. Electrical 10
      5.3.1. Operational ranges 10
      5.3.2. Current consumption 11
         5.3.2.1. DC-powered 11
         5.3.2.2. Battery-powered, variable voltage 11
         5.3.2.3. USB powered 11

6. Overview of platforms 13

7. Pin assignment 15
   7.1. Radio module interface 15
   7.2. Debug interface 20
   7.3. JTAG for ARM 21
   7.4. JTAG for AVR 22
   7.5. User Interface 22
   7.6. Jumper configuration 29

8. Board features 30
   8.1. Onboard sensors 31
      8.1.1. Temperature sensor 31
      8.1.2. Ambient light sensor 31
      8.1.3. Acceleration sensor 32
   8.2. Onboard Dataflash 32
   8.3. LEDs and buttons 33
      8.3.1. User LEDs 33
      8.3.2. User buttons 33
   8.4. USB interface 34
      8.4.1. Native USB only for ARM based radio modules 34
      8.4.2. USB serial for AVR based radio modules 34
   8.5. Ethernet 36
   8.6. Power supply 37
   8.7. Supervisor 38
   8.8. Current measurement 40
   8.9. USB supply voltage monitoring 40

8.10. Battery supply voltage monitoring 40

9. Case variants 41

10. Programming 42
10.1. Requirements (HW/SW) 42
10.2. Source code and compiler toolchain 42
10.3. Programming and adapter selection 42
10.4. Software programming model 42
10.4.1. Enabling the reset supervisor 43
10.4.2. Initialize and use I^2C devices 43
10.4.3. Using the USB interface 44
10.4.4. Measuring the battery voltage 45
10.4.5. Accessing the external flash 47
10.4.6. Initialize and use the Ethernet transceiver 49
10.4.7. Minimize device power consumption 49

11. Ordering information 52

12. Revision notes 53

References 54

Document history

Date	Version	Description
2011-04-15	1.0	Initial version
2011-07-15	1.1	Update feature list
2011-09-28	1.2	Update case variants Addition of signal descriptions
2014-04-10	1.3	Addition of OEM radio modules Update feature list Update hardware selection example Update operation conditions Update operational ranges Update overview of platforms Update radio module interface Update supervisor section Minor description changes Dataflash replacement

Abbreviations

Abbreviation	Description
ADC	**A**nalog to **D**igital **C**onverter
ARM	**A**dvanced **R**ISC **M**achine. A kind of processor architecture.
AVR	Names a family of microcontrollers from Atmel.
BOD	**B**rown**o**ut-**D**etection
CE	**C**onsumer **E**lectronics
DBGU	**D**e**b**u**g** **U**nit. An UART dedicated to print debug traces – available on ARM microcontrollers only.
EMAC	**E**thernet **M**edia **A**ccess **C**ontroller
FCC	**F**ederal **C**ommunications **C**ommission
FTDI	USB to serial converter from FTDI
GPIO	**G**enerals **P**urpose **I**nput **O**utput
I^2C	**I**nter-**I**ntegrated **C**ircuit, another name for TWI.
LDO	**L**ow-**D**ropout (**R**egulator)
JTAG	**J**oint **T**est **A**ction **G**roup, defines a standardized interface for programming and debugging microcontrollers.
µC, MCU	**M**icro **C**ontroller (**U**nit)
PCBA	**P**rinted **C**ircuit **B**oard **A**ssembled
PHY	**Phy**sical layer, refers to the lowest possible layer in a layered communication model
RF	**R**adio **F**requency
RMII	**R**educed **M**edia **I**ndependent **I**nterface
SMT	**S**urface **M**ount **T**echnology
SPI	**S**erial **P**eripheral **I**nterface
THT	**T**hrough-**H**ole **T**echnology
Transceiver	**Trans**mitter / Re**ceiver**
TWI	**T**wo-**W**ire Serial **I**nterface
U[S]ART	**U**niversal [**S**ynchronous/]**A**synchronous **R**eceiver **T**ransmitter
USB	**U**niversal **S**erial **B**us

1. Overview

The deRFnode and deRFgateway are demonstration and application platforms for the AVR and ARM based dresden elektronik radio modules. They support AVR and ARM programming and communication over Serial, USB and Ethernet interface. Assembled environmental sensors supplies data for a huge bandwidth of user defined applications.

2. Application

The main applications for the deRFgateway platform are:

- Coordinator and Router device for IEEE 802.15.4 compliant networks
- 6loWPAN nodes
- ZigBee®
- Gateway between IEEE 802.15.4 and IEEE 802.3
- Wireless Sensor Networks

The main applications for the deRFnode platform are:

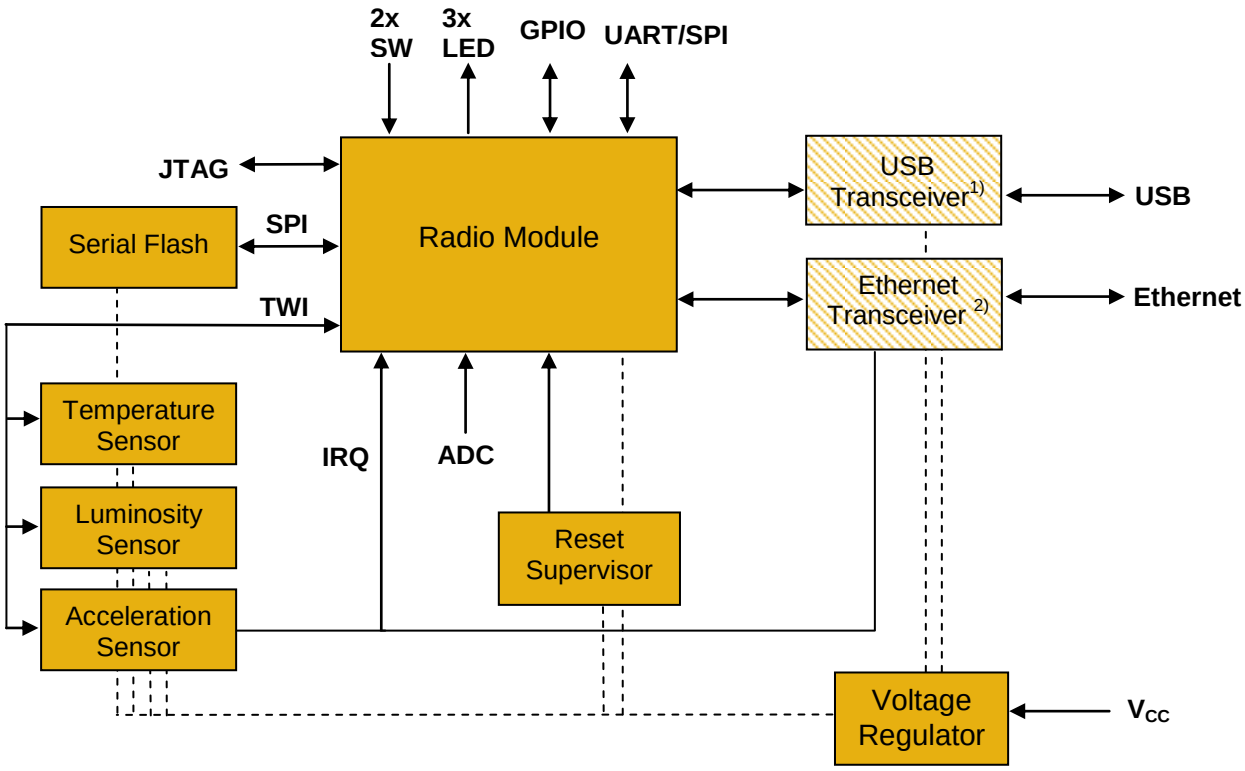
- Stand-Alone End device for a IEEE 802.15.4 compliant network
- applicable as coordinator for small networks
- battery powered applications with a lifetime of several years
- 6loWPAN nodes
- ZigBee

3. Features

The main features of deRFnode and deRFgateway are:

- Compact size: 69 x 75 x 30 mm
- Supports pluggable AVR or ARM based dresden elektronik radio modules
- Full Speed USB and 10/100 Mb/s Ethernet interface
- JTAG interface for AVR or ARM
- Serial debug interface
- Onboard Sensors: acceleration, temperature and luminosity
- Onboard 4Mbit Serial Flash
- Power Supply over USB, battery and 5V DC-Plug possible
- 2x buttons and 3x LEDs (free programmable)
- User interface with all important signals (2x17 pins connector)
- Switchable reset supervisor. Triggers on V_{CC}<2.4V (deRFnode for AVR) respective on V_{CC}<3.0V (deRFnode and deRFgateway for ARM).
- CE for deRFnode
- CE pending for deRFgateway
  (The deRFgateway is intended for laboratory, development, demonstration or evaluation purposes only)

3.1. Block diagram



[1] Full Speed; optionally (only on deRFnode-series with AVR, otherwise included in MCU)

[2] 10/100 Mb/s; on deRFgateway only

Figure 1: block diagram deRFnode/deRFgateway-series

3.2. Hardware selection table

From the electrical view, all deRF-radio modules may be combined with all deRFnode and deRFgateway baseboards. However, not every peripheral available on the baseboard is usable or accessible by the radio module due to routing constraints respective missing MCU features.

All supported radio module platforms and variants are listed in **Table 1**.

Table 1: Available board and radio module combinations

Type code	optimized for radio modules
plain variant	
deRFnode-1TNP2-00N00	deRFarm7 series deRFsam3 series
deRFnode-2TNP2-00N00	deRFmega128 series deRFmega256 series
deRFgateway-1TNP2-00N00	deRFarm7 series

Every variant is specifiable by a type code, which contains important key features of the platform. **Table 2** describes this code.

Table 2: Type code description

Hardware selection table												
deRFnode	-	x	x	x	x	x	-	x	x	x	x	x
deRFgateway	-	x	x	x	x	x	-	x	x	x	x	x
platform properties												
		1	Native USB									
		2	USB over FTDI									
			T	THT								
				N	None (no delivered radio module)							
					P	Plain (only PCBA)						
						2	Revision 2					
							0	0	N	0	0	w/o radio module

3.3. Feature list

This section gives an overview of the supported radio modules and features in combination with deRFnode and deRFgateway.

Table 3: Feature list (x = full support, o = limited support)

Platform	**Radio module**		**Supported features**						
		Native USB	Serial USB	Ethernet	Sensors	LEDs and Buttons	Serial Flash	USB powered	Low Power LDO
deRFnode									
1TNP2-00N00	deRFarm7 Series	**x**			**x**	**x**	**x**	**x**	
	deRFsam3 Series	**x**			**o**	**o**		**x**	
2TNP2-00N00	deRFmega128 Series		**x**		**x**	**x**	**x**	**x**	**x**
	deRFmega256 Series		**x**		**x**	**x**	**x**	**x**	**x**
deRFgateway									
1TNP2-00N00	deRFarm7 Series	**x**		**x**	**x**	**x**	**x**	**x**	

The radio modules of deRFsam3 Series do not support the onboard acceleration sensor, LED1, LED3, and Button1.

4. Hardware selection examples

The growing number of platform and radio module combinations makes it difficult and complex for the customer, to make the right choice of hardware depending on the customer application. The following section should give some examples for different applications.

4.1. ZigBee sensor network with battery powered nodes

Application:
A small network with three end-devices (nodes) should measure and transmit the temperature sensor data every minute to one network coordinator (master). The coordinator is DC or USB powered all the time. The nodes are battery powered and are sleeping all the time, except when they should measure and transmit the sensor data. The sensor network protocol is based on the ZigBee Stack by Atmel.

Required components:

Platforms:	deRFnode-2TNP2-00N00
Radio modules:	deRFmega128-22A00
Software:	based on BitCloud (Atmel's ZigBee Stack)

Development Kit:

deRFdevelopmentKit ZigBee 2,4 GHz
Order code BN-032515
Available at https://shop.dresden-elektronik.de

4.2. 6LoWPAN tree-network application

Application:
A wireless network that can be monitored and controlled via Ethernet by Remote Access. The nodes have their own unique MAC-address and a user-defined IP-address. They can be equipped with sensors and/or actuators, which read out sensor data and/or switch on/off a relay.

Required components:

Platforms:	deRFnode-2TNP2-00N00
	deRFgateway-1TNP2-00N00
Radio modules:	deRFmega128-22A00
	deRFarm7-25A00
Software:	6LoWPAN-Stack by dresden elektronik

Development Kit:

deRFdevelopmentKit 6LoWPAN 2,4 GHz
Order code BN-032518
Available at https://shop.dresden-elektronik.de

4.3. Point-to-Point connection for simple applications

Application:
The simplest network is a point-to-point connection between two devices. There is no need to use a complex protocol.

Required components:

Platforms:	deRFnode-2TNP2-00N00
Radio modules:	deRFmega128-22A00
Software:	Wireless UART based on Atmel's MAC-Stack

5. Technical data

5.1. Mechanical

Table 4: Mechanical data

Mechanical	
baseboard including radio module	
Size of PCBA (L x W x H)	69 x 75 x 30 mm

5.2. Operation conditions

The recommended operating conditions are as follows:

External supply voltage: VCC = 5.0V ± 0.3V DC (via AC/DC converter, USB or battery)
Internal supply voltage: VCC = 3.3V ± 0.3V DC
Temperature: T = -40°C to +85°C

5.3. Electrical

5.3.1. Operational ranges

Since the voltage regulators threshold is fixed to 3.3V DC, operation is uncritical as long as input voltage is above 3.3V. Below, operation is not recommended since assembled components (MCU, Flash, EMAC, I^2C-Sensors) will start to fail. The probability that they do grows, the lower the voltage is. For concrete working voltage ranges please refer to the table below as well as the respective components datasheets.

To avoid unstable behaviour, the board supplies a reset supervisor which drives a pin low, if the input voltage sinks below 2.4V DC (deRFnode/gateway for AVR) respective 3.0V DC (deRFnode/-gateway for ARM). This pin is routed to the radio module MCUs reset entry (Pin5). To enable a too low voltage causing a MCU reset, JP4 must be closed. On ARM-MCUs the reset supervisor must be explicitly enabled (see **Section 8.7**).

Table 5: Operational ranges

Device	**Remark**	**Required operational voltage range**		**Current consumption**		
		Min	*Max*	*Min*	*Typ*	*Max*
AT91SAM7X512	on deRFarm7 radio modules	3.0V	3.6V	60µA	90mA[2)]	200mA[2)]
AT86RF231 and AT86RF212	any Atmel radio transceiver used on deRFarm7 radio modules	1.8V	3.6V	≤0.2µA	~12mA[1)]	<25mA
Atmega128RFA1	on deRFmega128 radio modules	1.8V	3.6V	20nA	12,5mA[1,2)]	--- [2)]
Atmega256RFR2	on deRFmega256 radio modules	1.8V	3.6V	20nA	12,5mA[1,2)]	--- [2)]

deRFmega128-22T13 or deRFmega256-23T13	Atmega128RFA1 or Atmega256RFR2 with onboard Front-End (PA + LNA)	2.0V	3.6V	~1µA	22.5mA [1,2)]	<233mA
DP83848C	on deRFgateway for ARM only	3.0V	3.3V	14mA	---	92mA
BMA150	acceleration sensor	2.4V	3.6V	1µA	200µA	290µA
ISL29020	luminosity sensor	1.7V	3.6V	500nA	---	65µA
TMP102AIDRLT	temperature sensor	TBD	TBD	TBD	TBD	TBD
AT25DF041A-SSHF	4Mbit serial Flash	2.3V	3.6V	25µA	10mA	20mA
M25P40-VMN6TPB	4Mbit serial Flash	2.3V	3.6V	10µA	8mA	15mA
FT245RL	USB for AVRs	4.0V	5.25V	50µA	15mA	24mA

[1)] radio transceiver in listening state

[2)] depends on external load

5.3.2. Current consumption

Test conditions: T = 25°C, Firmware executed from Flash, no external cabling (i.e. Level Shifter, JTAG) unless stated otherwise.

5.3.2.1. DC-powered

The used AC/DC converter has an output voltage of 5VDC.

Table 6: HW Setup 1

Hardware setup (Condition: V_{DC} = 5VDC)		**Working**		
	Sleep [1)]	*Idle [2)]*	*Typ [3)]*	*Max [4)]*
deRFgateway-1TNP2-00N00 + deRFarm7	24mA	97mA [5)]	161mA [6)]	<200mA
deRFnode-1TNP2-00N00 + deRFarm7	250µA	37mA	41mA	<80mA
deRFnode-2TNP2-00N00 + deRFarm7	5mA	34mA	38mA	<80mA
deRFnode-2TNP2-00N00 + deRFmega128	10µA	10mA	20mA	<40mA

[1)] … [6)] … see **Table 7**.

5.3.2.2. Battery-powered, variable voltage

When battery powered, the current consumption does not significantly differ from the values given above, only if using an AVR-based MCU, the current consumption sinks slightly. Remark the notes on working voltage above.

5.3.2.3. USB powered

When USB-powered, the current consumption increases due to USB transceiver activity:

Table 7: HW Setup 2

Hardware setup (Condition: V_{USB} = 5VDC)		**Working**		
	Sleep [1)]	*Idle* [2)]	*Typ* [3)]	*Max* [4)]
deRFgateway-1TNP2-00N00 + deRFarm7	24mA	97mA [5)]	166mA [6)]	<200mA
deRFnode-1TNP2-00N00 + deRFarm7	250µA	37mA	41mA	<85mA
deRFnode-2TNP2-00N00 + deRFarm7	14mA	45mA	49mA	<85mA
deRFnode-2TNP2-00N00 + deRFmega128	12mA	20mA	30mA	<50mA

[1)] … peripherals and MCU put to sleep as far as possible

[2)] … all peripheral initialized but not accessed

[3)] … typical application scenario (sensors accessed once each second, Transceiver off)

[4)] … theoretical value, every onboard peripheral accessed

[5)] … Ethernet cable not plugged

[6)] … Ethernet cable plugged, 100Mbps Link established

6. Overview of platforms

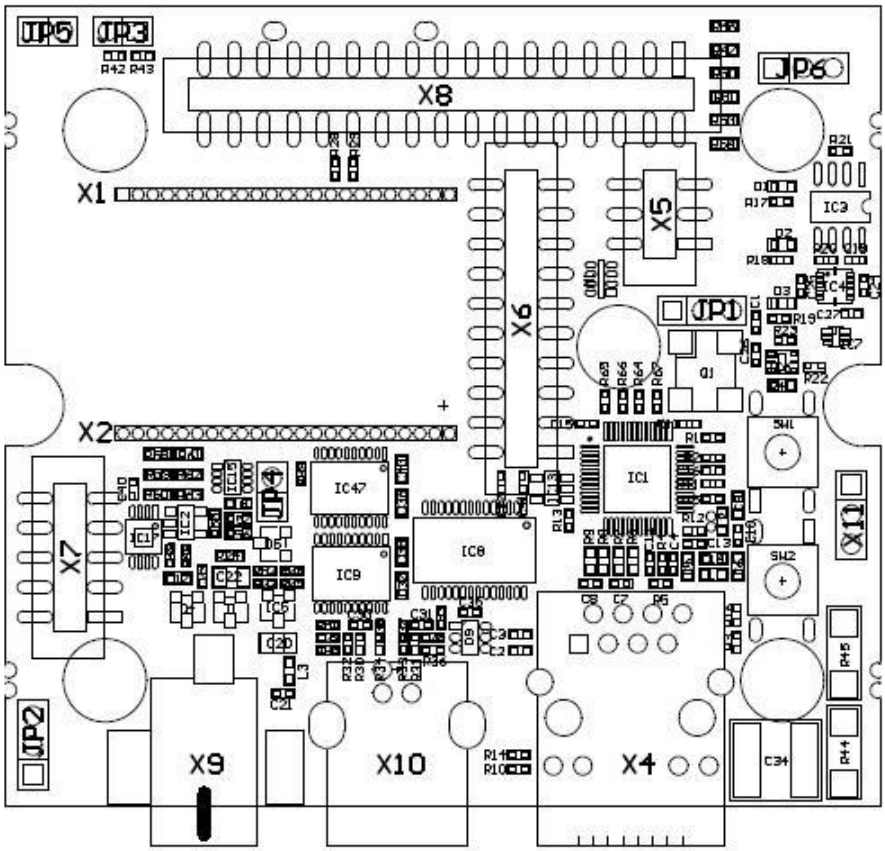


Figure 2: Overlay of deRFnode and deRFgateway

Table 8: Overview headers

Header	
Name	*Description*
X1	Radio module interface, 2x23 Pin Radio module header – only available for pluggable platform variant
X2	
X3	3 Pin Button extension – only available for case variant
X4	Ethernet RJ45 socket
X5	Debug interface, 6 Pin Debug header
X6	JTAG for ARM, 20 Pin JATG header for ARM programmer
X7	JTAG for AVR, 10 Pin JTAG header for AVR programmer
X8	User Interface, 34 Pin user header
X9	5 VDC connector for power supply
X10	USB type B plug for power supply and data exchange

The pin number ‘one’ is marked with a triangle on all headers in **Figure 3**.

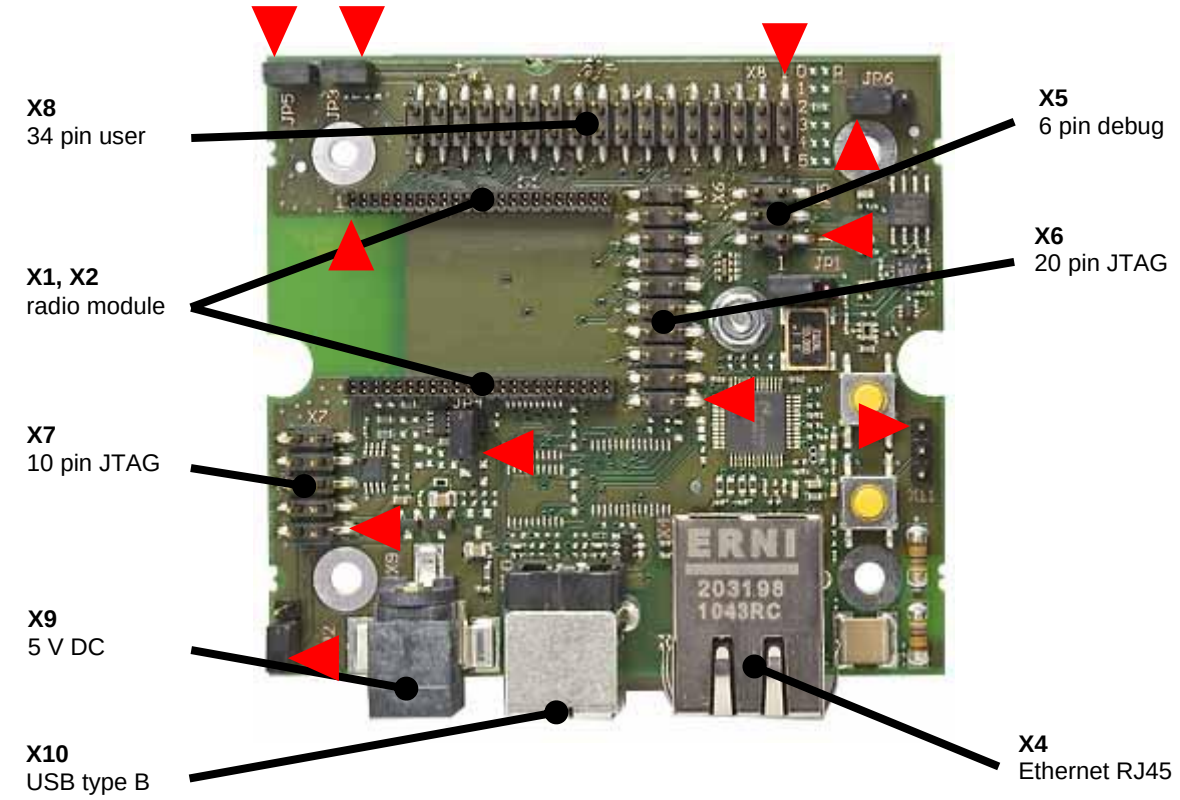


Figure 3: deRFgateway-1TNP2

Figure 4: deRFnode-1TNP2

Figure 5: deRFnode-2TNP2

7. Pin assignment

This section describes the available headers on the deRFnode and deRFgateway platforms as summarized in **Table 8**.

7.1. Radio module interface

The deRFnode and deRFgateway will support all dresden elektronik radio modules. Depending on the radio module and the platform, some features will not be supported. The details of radio module specific signals are available in the associated user manuals.

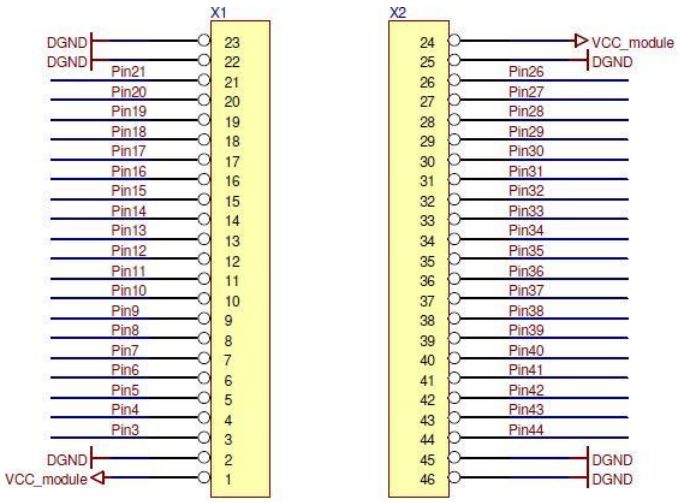


Figure 6: Header for radio modules

The next two tables give an overview of the radio module signals. **Table 9** shows the header pins, signal names and MCU ports of the deRFmega128 and deRFmega256 radio module series of dresden elektronik, **Table 10** for the deRFarm7 radio module series, **Table 11** for deRFsam3 radio module series and **Table 12** only for deRFsam3-23T02-3R and deRFsam3-23T09-3R

Table 9: Pin assignment for deRFmega128-series and deRFmega256-series

Pin assignment					
Header pin	*Signal name*	*µC-Port (deRFmega128/256)*	*Header pin*	*Signal name*	*µC-Port (deRFmega128/256)*
1	-	VCC	24	-	VCC
2	-	GND	25	-	GND
3	Pin3	AREF	26	Pin26	PE0/RXD0/PCINT8
4	Pin4	PG1/DI1	27	Pin27	PD2/RXD1/INT2
5	Pin5	RSTN	28	Pin28	PE1/TXD0
6	Pin6	PG2	29	Pin29	PD6/T1
7	Pin7	PD0/SCL/INT0	30	Pin30	PE2/XCK0/AIN0
8	Pin8	PG5/OC0B	31	Pin31	PE3/OC3A/AIN1
9	Pin9	PD1/SDA/INT1	32	Pin32	PD4/ICP1
10	Pin10	PD3/TXD1/INT3	33	Pin33	PE4/OC3B/INT4
11	Pin11	PD7/T0	34	Pin34	PF0/ADC0
12	Pin12	PD5/XCK1	35	Pin35	PE5/OC3C/INT5
13	Pin13	PB1/SCK/PCINT1	36	Pin36	PF1/ADC1
14	Pin14	CLKI	37	Pin37	PE6/T3/INT6
15	Pin15	PB2/MOSI/PCINT2/PDI	38	Pin38	PF4/ADC4/TCK
16	Pin16	PB0/SSN/PCINT0	39	Pin39	PE7/ICP3/CLKO/INT7
17	Pin17	PB3/MISO/PCINT3/PDO	40	Pin40	PF5/ADC5/TMS
18	Pin18	PB6/OC1B/PCINT6	41	Pin41	PF2/ADC2
19	Pin19	PB4/OC2/PCINT4	42	Pin42	PF6/ADC6/TDO
20	Pin20	PB7/OC0A/OC1C/PCINT7	43	Pin43	RSTON
21	Pin21	PB5/OC1A/PCINT5	44	Pin44	PF7/ADC7/TDI
22	-	GND	45	-	GND
23	-	GND	46	-	GND

Table 10: Pin assignment for deRFarm7-series

Pin assignment					
Header pin	*Signal name*	*µC-Port (deRFarm7)*	*Header pin*	*Signal name*	*µC-Port (deRFarm7)*
1	-	VCC	24	-	VCC
2	-	GND	25	-	GND
3	Pin3	ADVREF	26	Pin26	PA27/DRXD/PCK3
4	Pin4	USBDM	27	Pin27	PA0/RXD0
5	Pin5	RSTN	28	Pin28	PA28/DTXD
6	Pin6	PB3/ETX1	29	Pin29	PA4/CTS0/SPI1_NPCS3
7	Pin7	PA11/TWCK	30	Pin30	PB9/EMDIO
8	Pin8	PB26/TIOB1/RI1	31	Pin31	PB21/PWM2/PCK1
9	Pin9	PA10/TWD	32	Pin32	USBDP
10	Pin10	PA1/ TXD0	33	Pin33	PB19/PWM0/TCLK1
11	Pin11	PB25/TIOA1/DTR1	34	Pin34	PB27/TIOA2/PWM0/AD0
12	Pin12	PB2/ETX0	35	Pin35	PA14/SPI0_NPCS2/IRQ1
13	Pin13	PA18/SPI0_SPCK	36	Pin36	PB28/TIOB2/PWM1/AD1
14	Pin14	PA3/RTS0/SPI1_NPCS2	37	Pin37	PB5/ERX0
15	Pin15	PA17/SPI0_MOSI	38	Pin38	TCK
16	Pin16	PB0/ETXCK/EREFCK	39	Pin39	PB7/ERXER
17	Pin17	PA16/SPI0_MISO	40	Pin40	TMS
18	Pin18	PB8/EMDC	41	Pin41	PB1/ETXEN
19	Pin19	PB6/ERX1	42	Pin42	TDO
20	Pin20	PB18/EF100/ADTRG	43	Pin43	JTAGSEL
21	Pin21	PB15/ERXDV/ECRSDV	44	Pin44	TDI
22	-	GND	45	-	GND
23	-	GND	46	-	GND

Table 11: Pin assignment for deRFsam3-series 13T02 / 23T02-2

Pin assignment					
Header pin	*Signal name*	*µC-Port (deRFsam3)*	*Header pin*	*Signal name*	*µC-Port (deRFsam3)*
1	-	VCC	24	-	VCC
2	-	GND	25	-	GND
3	Pin3	ADVREF	26	Pin26	PA9/URXD0/NPCS1
4	Pin4	PB10/DDM	27	Pin27	PB2/URXD1/NPCS2/AD6
5	Pin5	RSTN	28	Pin28	PA10/UTXD0/NPCS2
6	Pin6	PA7/RTS0/PWMH3/XIN32	29	Pin29	PA8/CTS0/ADTRG/XOUT32
7	Pin7	PA4/TWCK0/TCLK0	30	Pin30	PB0/PWMH0/AD4
8	Pin8	NC	31	Pin31	PB1/PWMH1/AD5
9	Pin9	PA3/TWD0/NPCS3	32	Pin32	PB11/DDP
10	Pin10	PB3/UTXD1/PCK2/AD7	33	Pin33	NC
11	Pin11	NC	34	Pin34	PA18/RD/PCK2/AD1
12	Pin12	NC	35	Pin35	NC
13	Pin13	PA2/PWMH2/SCK0	36	Pin36	PA19/RK/PWML0/AD2
14	Pin14	NC	37	Pin37	NC
15	Pin15	PA6/TXD0/PCK0	38	Pin38	PB7/TXK/SWCLK
16	Pin16	NC	39	Pin39	NC
17	Pin17	PA5/RXD0/NPCS3	40	Pin40	PB6/TMS/SWDIO
18	Pin18	NC	41	Pin41	PA20/RF/PWML1/AD3
19	Pin19	NC	42	Pin42	PB5/TDO/TWCK1
20	Pin20	NC	43	Pin43	JTAGSEL
21	Pin21	NC	44	Pin44	PB4/TDI/TWD1
22	-	GND	45	-	GND
23	-	GND	46	-	GND

Table 12: Pin assignment for deRFsam3-23T09-3R

Pin assignment					
Header pin	*Signal name*	*µC-Port (deRFsam3)*	*Header pin*	*Signal name*	*µC-Port (deRFsam3)*
1	-	VCC	24	-	VCC
2	-	GND	25	-	GND
3	Pin3	ADVREF	26	Pin26	PA9/URXD0/NPCS1
4	Pin4	PB10/DDM	27	Pin27	PB2/URXD1/NPCS2/AD6
5	Pin5	RSTN	28	Pin28	PA10/UTXD0/NPCS2
6	Pin6	PA7/RTS0/PWMH3/XIN32	29	Pin29	PA8/CTS0/ADTRG/XOUT32
7	Pin7	PB5/TDO/TWCK1	30	Pin30	PB0/PWMH0/AD4
8	Pin8	NC	31	Pin31	PB1/PWMH1/AD5
9	Pin9	PB4/TDI/TWD1	32	Pin32	PB11/DDP
10	Pin10	PB3/UTXD1/PCK2/AD7	33	Pin33	NC
11	Pin11	NC	34	Pin34	PA18/RD/PCK2/AD1
12	Pin12	NC	35	Pin35	NC
13	Pin13	PA2/PWMH2/SCK0	36	Pin36	PA19/RK/PWML0/AD2
14	Pin14	NC	37	Pin37	NC
15	Pin15	PA6/TXD0/PCK0	38	Pin38	PB7/TXK/SWCLK
16	Pin16	PA3/TWD0/NPCS3	39	Pin39	NC
17	Pin17	PA5/RXD0/NPCS3	40	Pin40	PB6/TMS/SWDIO
18	Pin18	NC	41	Pin41	PA20/RF/PWML1/AD3
19	Pin19	NC	42	Pin42	PB5/TDO/TWCK1
20	Pin20	PA0/PWMH0/TIOA0	43	Pin43	JTAGSEL
21	Pin21	NC	44	Pin44	PB4/TDI/TWD1
22	-	GND	45	-	GND
23	-	GND	46	-	GND

7.2. Debug interface

The debug header may be used for device interconnecting via USART, like on a PC. Remember that a level shifter between TTL and RS232 may be required.

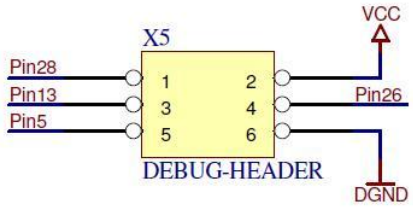


Figure 7: Debug header

The following table shows the signal description.

Table 13: Debug Header Pin assignment

Pin assignment					
Header pin	*Signal name*	*Function*	*Header pin*	*Signal name*	*Function*
1	Pin28	TXD (UART0/DBGU)	2	-	VCC
3	Pin13	SCK	4	Pin26	RXD (UART0/DBGU)
5	Pin5	RSTN	6	-	GND

7.3. JTAG for ARM

The header layout conforms to the 20-pin assignment traditionally used for ARM MCUs.

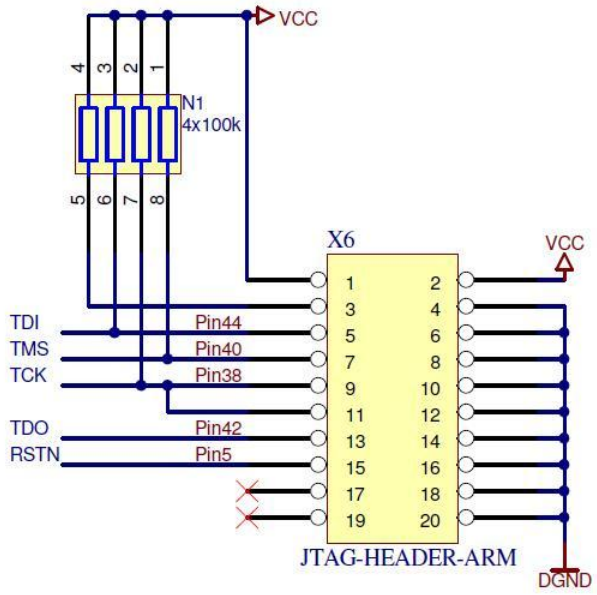


Figure 8: JTAG for ARM header

The following table shows the signal description.

Table 14: JTAG for ARM header pin assignment

Pin assignment					
Header pin	*Signal name*	*Function*	*Header pin*	*Signal name*	*Function*
1	-	VCC	2	-	VCC
3	-	100K Pullup	4	-	GND
5	Pin44	TDI, 100K Pullup	6	-	GND
7	Pin40	TMS, 100K Pullup	8	-	GND
9	Pin38	TCK, 100K Pullup	10	-	GND
11	Pin38	RTCK	12	-	GND
13	Pin42	TDO	14	-	GND
15	Pin5	RSTN	16	-	GND
17	-	N/C	18	-	GND
19	-	N/C	20	-	GND

7.4. JTAG for AVR

The header layout conforms to the 10-pin assignment used usually for AVR.

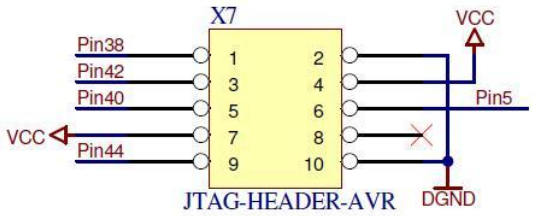


Figure 9: JTAG AVR header

The following table shows the signal description.

Table 15: JTAG for AVR header pin assignment

Pin assignment					
Header pin	*Signal name*	*Function*	*Header pin*	*Signal name*	*Function*
1	Pin38	TCK	2	-	GND
3	Pin42	TDO	4	-	VCC
5	Pin40	TMS	6	Pin5	RSTN
7	-	VCC	8	-	N/C
9	Pin44	TDI	10	-	GND

7.5. User Interface

The User Interface header provides access to a series of IO port pins.

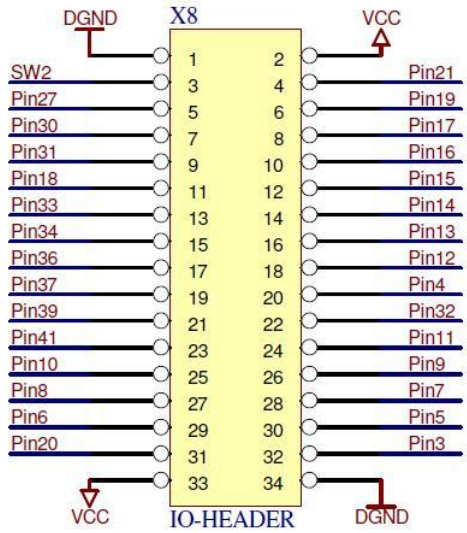


Figure 10: User header

The following tables show the signal description. The internal use shows if a signal on user interface is still **used** by internal peripherals or **free** usable.

Table 16: User interface header pin assignment for plugged deRFmega128 and deRFmega256

Pin assignment			**Internal use**		
Header pin	*Signal name*	*Function deRFmega128 deRFmega256*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
1	-	GND	Not recommended	-	-
2	-	VCC		-	-
3	SW2	-		Button SW2	
4	Pin21	PB5/OC1A/PCINT5		Free	Serial USB
5	Pin27	PD2/RXD1/INT2		Free	Serial USB
6	Pin19	PB4/OC2/PCINT4		Free	Serial USB
7	Pin30	PE2/XCK0/AIN0		Free	Serial USB
8	Pin17	PB3/MISO/PCINT3/PDO		Flash	Flash
9	Pin31	PE3/OC3A/AIN1		LED2	LED2
10	Pin16	PB0/SSN/PCINT0		Free	Serial USB
11	Pin18	PB6/OC1B/PCINT6		Free	Serial USB
12	Pin15	PB2/MOSI/PCINT2/PDI		Flash	Flash
13	Pin33	PE4/OC3B/INT4		LED3	LED3
14	Pin14	CLKI		Clock Input	
15	Pin34	PF0/ADC0		Battery voltage monitoring selectable with jumper JP3	
16	Pin13	PB1/SCK/PCINT1		Flash	Flash
17	Pin36	PF1/ADC1		USB voltage monitoring	
18	Pin12	PD5/XCK1		Free	Serial USB
19	Pin37	PE6/T3/INT6		Free	Serial USB
20	Pin4	PG1/DIG1		Not available	Free
21	Pin39	PE7/ICP3/CLKO/INT7		Free	Serial USB
22	Pin32	PD4/ICP1		Not available	Free
23	Pin41	PF2/ADC2		Free	Serial USB
24	Pin11	PD7/T0		Interrupt output of acceleration sensor or button SW2 selectable with jumper JP1	

Header pin	*Signal name*	*Function deRFmega128 deRFmega256*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
25	Pin10	PD3/TXD1/INT3	Not recommended	Free	Serial USB
26	Pin9	PD1/SDA/INT1		Acceleration, temperature and luminosity sensor	
27	Pin8	PG5/OC0B		LED1	LED1
28	Pin7	PD0/SCL/INT0		Acceleration, temperature and luminosity sensor	
29	Pin6	PG2		Free	Serial USB
30	Pin5	RSTN		Reset	
31	Pin20	PB7/OC0A/OC1C/PCINT7		Button SW1 selectable with jumper JP6	
32	Pin3	AREF		Analog reference voltage	
33	-	VCC		-	-
34	-	GND		-	-

Table 17: User interface header pin assignment for plugged deRFarm7

Pin assignment			**Internal use**		
Header pin	*Signal name*	*Function deRFarm7*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
1	-	GND	-	-	Not recommended
2	-	VCC	-	-	
3	SW2		Used	Used	
4	Pin21	PB15/ERXDV/ECRSDV	Ethernet Phy	Free	
5	Pin27	PA0/RXD0	Free	Free	
6	Pin19	PB6/ERX1	Ethernet Phy	Free	
7	Pin30	PB9/EMDIO	Ethernet Phy	Free	
8	Pin17	PA16/SPI0_MISO	Flash	Flash	
9	Pin31	PB21/PWM2/PCK1	LED2	LED2	
10	Pin16	PB0/ETXCK/EREFCK	Ethernet Phy	Free	
11	Pin18	PB8/EMDC	Ethernet Phy	Free	
12	Pin15	PA17/SPI0_MOSI	Flash	Flash	
13	Pin33	PB19/PWM0/TCLK1	LED3	LED3	

Header pin	*Signal name*	*Function deRFarm7*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
14	Pin14	PA3/RTS0/SPI1_NPCS2	Button SW1 selectable with jumper JP6		Not recommended
15	Pin34	PB27/TIOA2/PWM0/AD0	Battery voltage monitoring selectable with jumper JP3		
16	Pin13	PA18/SPI0_SPCK	Flash	Flash	
17	Pin36	PB28/TIOB2/PWM1/AD1	USB voltage monitoring		
18	Pin12	PB2/ETX0	Ethernet Phy	Free	
19	Pin37	PB5/ERX0	Ethernet Phy	Free	
20	Pin4	USBDM	Native USB		
21	Pin39	PB7/ERXER	Ethernet Phy	Free	
22	Pin32	USBDP	Native USB		
23	Pin41	PB1/ETXEN	Ethernet Phy	Free	
24	Pin11	PB25/TIOA1/DTR1	Interrupt output of acceleration sensor or button SW2 selectable with jumper JP1		
25	Pin10	PA1/ TXD0	Free	Free	
26	Pin9	PA10/TWD	Acceleration, temperature and luminosity sensor		
27	Pin8	PB26/TIOB1/RI1	LED1	LED1	
28	Pin7	PA11/TWCK	Acceleration, temperature and luminosity sensor		
29	Pin6	PB3/ETX1	Ethernet Phy	Free	
30	Pin5	RSTN	Reset		
31	Pin20	PB18/EF100/ADTRG	Ethernet Phy	Free	
32	Pin3	ADVREF	Analog reference voltage		
33	-	VCC	-	-	
34	-	GND	-	-	

Table 18: User interface header pin assignment for plugged deRFsam3-13T02 / 23T02-2

Pin assignment			**Internal use**		
Header pin	*Signal name*	*Function deRFsam3*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
1	-	GND	Not recommended	-	Not recommended
2	-	VCC		-	
3	SW2	NC		NC	
4	Pin21	NC		NC	
5	Pin27	PB2/URXD1/NPCS2		Free	
6	Pin19	NC		NC	
7	Pin30	PB0/PWMH0/AD4		Free	
8	Pin17	PA5/RXD0/NPCS3		Flash	
9	Pin31	PB1/PWMH1/AD5		LED2	
10	Pin16	NC		NC	
11	Pin18	NC		NC	
12	Pin15	PA6/TXD0/PCK0		Flash	
13	Pin33	NC		NC	
14	Pin14	NC		NC	
15	Pin34	PA18/RD/PCK2/AD1		Battery voltage monitoring selectable with jumper JP3	
16	Pin13	PA2/PWMH2/SCK0		Flash	
17	Pin36	PA19/RK/PWML0/AD2		USB voltage monitoring	
18	Pin12	NC		NC	
19	Pin37	NC		NC	
20	Pin4	PB10/DDM		Native USB	
21	Pin39	NC		NC	
22	Pin32	PB11/DDP		Native USB	
23	Pin41	PA20/RF/PWML1/AD3		Free	
24	Pin11	NC		NC	
25	Pin10	PB3/UTXD1/PCK2/AD7		Free	
26	Pin9	PA3/TWD0/NPCS3		Acceleration, temperature and luminosity sensor	

Header pin	*Signal name*	*Function deRFsam3*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
27	Pin8	NC	Not recommended	NC	Not recommended
28	Pin7	PA4/TWCK0/TCLK0		Acceleration, temperature and luminosity sensor	
29	Pin6	PA7/RTS0/PWMH3/XIN32		Free	
30	Pin5	RSTN		Reset	
31	Pin20	NC		NC	
32	Pin3	ADVREF		Analog reference voltage	
33	-	VCC		-	
34	-	GND		-	

Table 19: User interface header pin assignment for plugged deRFsam3-23T09-3R

Pin assignment			**Internal use**		
Header pin	*Signal name*	*Function deRFsam3*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
1	-	GND	Not recommended	-	Not recommended
2	-	VCC		-	
3	SW2	NC		NC	
4	Pin21	NC		NC	
5	Pin27	PB2/URXD1/NPCS2		Free	
6	Pin19	NC		NC	
7	Pin30	PB0/PWMH0/AD4		Free	
8	Pin17	PA5/RXD0/NPCS3		Flash	
9	Pin31	PB1/PWMH1/AD5		LED2	
10	Pin16	PA3/TWD0/NPCS3		Free	
11	Pin18	NC		NC	
12	Pin15	PA6/TXD0/PCK0		Flash	
13	Pin33	NC		NC	
14	Pin14	NC		NC	
15	Pin34	PA18/RD/PCK2/AD1		Battery voltage monitoring selectable with jumper JP3	
16	Pin13	PA2/PWMH2/SCK0		Flash	

Header pin	*Signal name*	*Function deRFsam3*	*deRF gateway-1TNP2*	*deRF node-1TNP2*	*deRF node-2TNP2*
17	Pin36	PA19/RK/PWML0/AD2	Not recommended	USB voltage monitoring	Not recommended
18	Pin12	NC		NC	
19	Pin37	NC		NC	
20	Pin4	PB10/DDM		Native USB	
21	Pin39	NC		NC	
22	Pin32	PB11/DDP		Native USB	
23	Pin41	PA20/RF/PWML1/AD3		Free	
24	Pin11	NC		NC	
25	Pin10	PB3/UTXD1/PCK2/AD7		Free	
26	Pin9	PB4/TDI/TWD1/PWMH2		Acceleration, temperature and luminosity sensor	
27	Pin8	NC		NC	
28	Pin7	PB5/TDO/TWCK1/PWML0		Acceleration, temperature and luminosity sensor	
29	Pin6	PA7/RTS0/PWMH3/XIN32		Free	
30	Pin5	RSTN		Reset	
31	Pin20	PA0/PWMH0/TIOA0		Free	
32	Pin3	ADVREF		Analog reference voltage	
33	-	VCC		-	
34	-	GND		-	

7.6. Jumper configuration

The following table shows the possible jumper configurations.

Table 20: Jumper configuration

Pin assignment	
JP	*Function*
1	GPIO Input diversity (SW2 if pins 1-2 closed / acceleration sensor interrupt output pin if pins 2-3 closed / free usable pin on user interface connector if left open)
2	Power Supply Selection (Battery or DC / USB)
3	VBAT Monitor (closed=enabled)
4	Reset Supervisor (closed=enabled)
5	Current measurement of radio module
6	Button 1 routing depending on radio module (ARM or AVR)

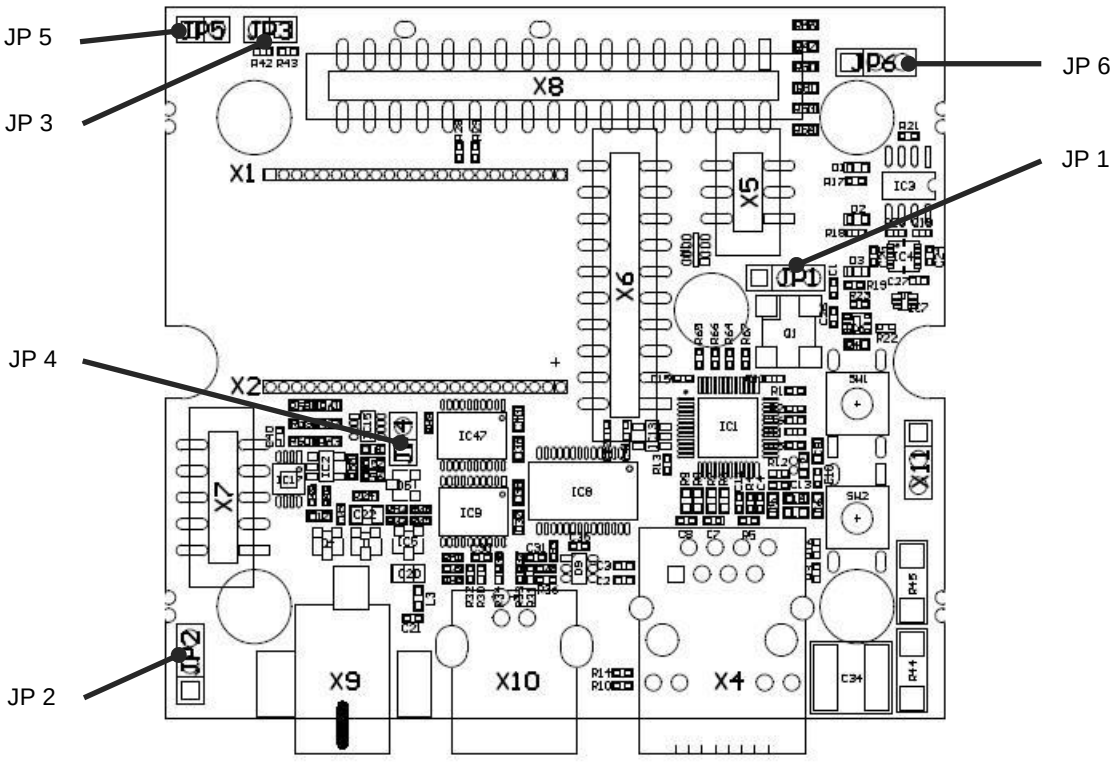


Figure 11: Jumper configuration

8. Board features

The deRFnode and deRFgateway platforms have a lot of available onboard features like three different sensors, user defined buttons and LEDs, USB and Ethernet interface, a supervisor and power supply monitoring.

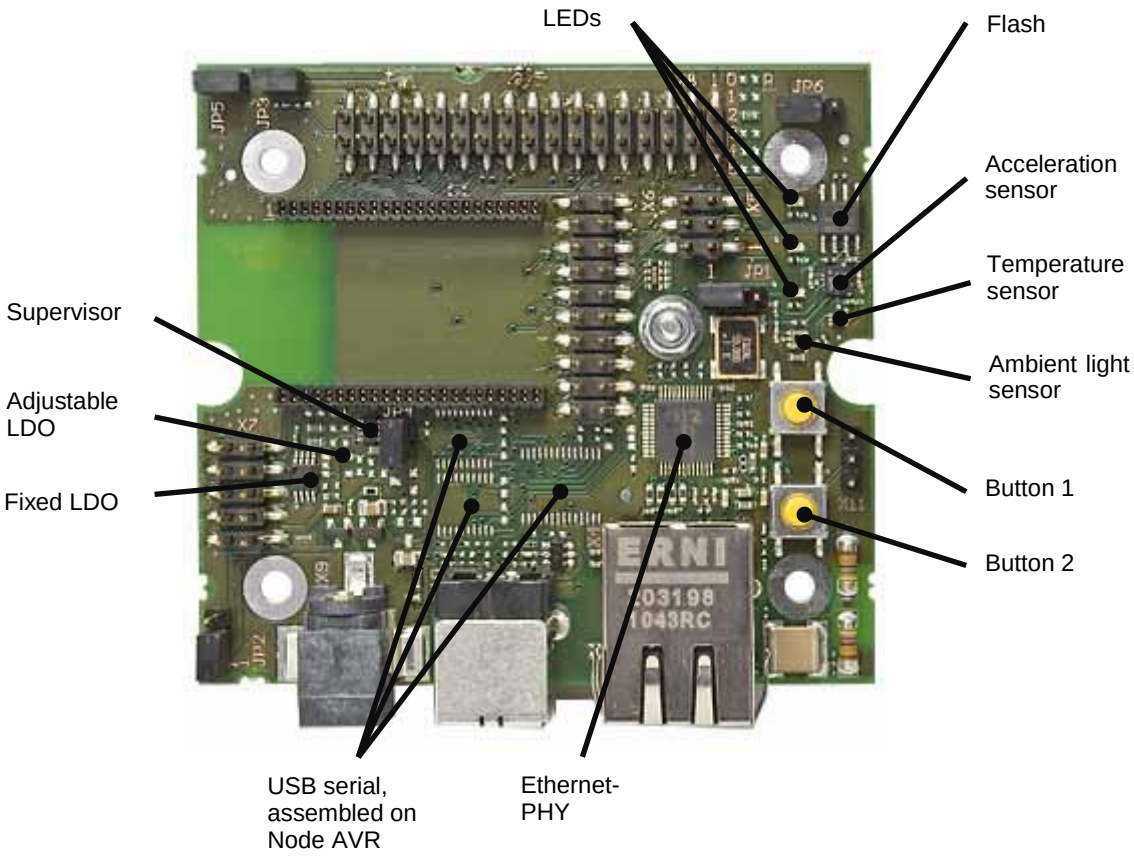


Figure 12: Board features

8.1. Onboard sensors

Both platforms are assembled with onboard sensors: temperature, ambient light and acceleration. All sensors are accessible over the two-wire-interface of deRFnode and deRFgateway. The device addresses are noted in the following subsections. The data and clock lines are equipped with 10k pull-up resistors [R22] and [R23].

TWI clock: Pin 7
TWI data: Pin 9

8.1.1. Temperature sensor

The temperature sensor TMP102AIDRLT communicates over two-wire-interface with the microcontroller of the radio module. Details of operation are described in the datasheet **[1]**.

TWI address: 1001 000 (R/W)
Write: R/W = 0
Read: R/W = 1

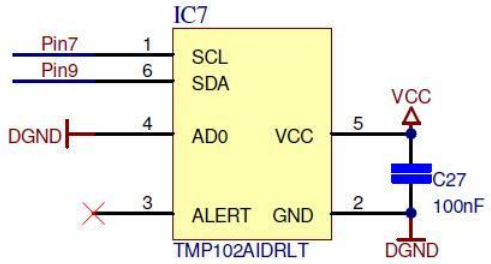


Figure 13: Temperature sensor TMP102AIDRLT

8.1.2. Ambient light sensor

The assembled ambient light sensor ISL29020IROZ-T7 communicates over two-wire-interface with the microcontroller of the radio module. Details of operation are described in the datasheet **[2]** .

TWI address: 1000 100 (R/W)
Write: R/W = 0
Read: R/W = 1

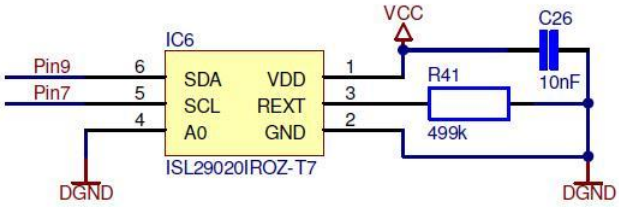


Figure 14: Ambient light sensor ISL29020IROZ-T7

8.1.3. Acceleration sensor

The acceleration sensor BMA150 communicates over two-wire-interface with the microcontroller of the radio module. Details of operation are described in the datasheet **[3]** .

The interrupt output of BMA150 could be connected with Pin 11 by setting the jumper [JP1] (pins 2 and 3).

TWI address: 0111 000 (R/W)
Write: R/W = 0
Read: R/W = 1

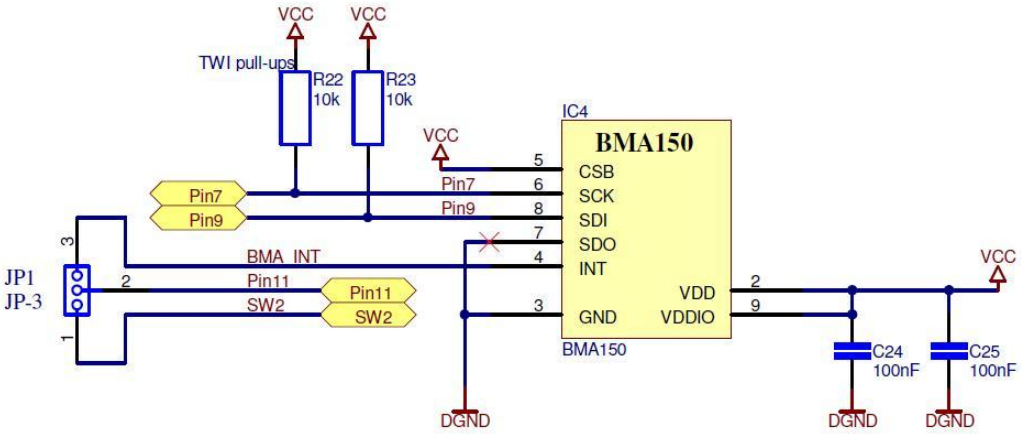
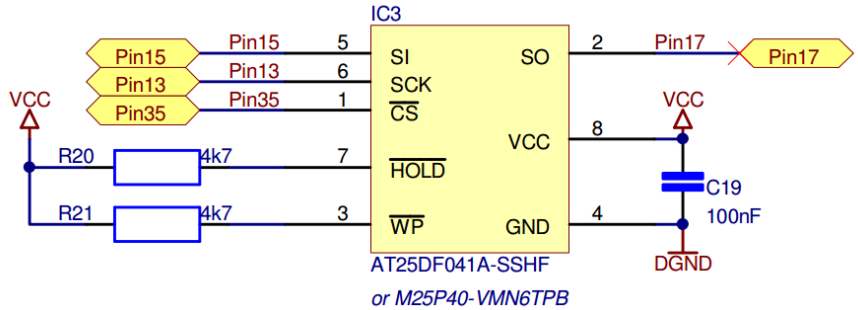


Figure 15: Acceleration sensor BMA150

8.2. Onboard Dataflash

The onboard dataflash communicates over the SPI interface with the microcontroller of the radio module. It can be used to persistently store custom data or to buffer firmware images when performing Over-the-Air-Updates. Older boards used Atmels AT25DF041, recent versions come with Micron M25P40. Details of operation are described in the respective datasheets **[4]**, **[5]** .

SPI clock: Pin 13
MOSI: Pin 15
MISO: Pin 17
SPI Select Pin 35



8.3. LEDs and buttons

The deRFnode and deRFgateway boards comprise three LEDs and two buttons, each user-defined controllable.

8.3.1. User LEDs

The three red LEDs are active-low and may be controlled by the radio module MCU.

LED1 [D1]: Pin 8
LED2 [D2]: Pin 31
LED3 [D3]: Pin 33

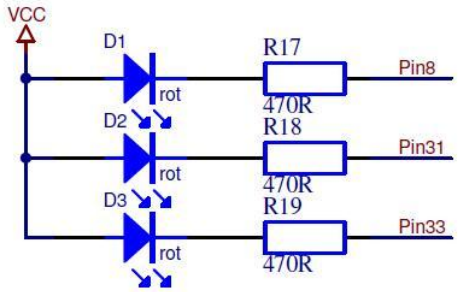


Figure 16: User LEDs

8.3.2. User buttons

The active-low buttons could be used for user defined inputs. The button [SW1] is controlled by two different pins, which are connected by assembling the 0 ohms resistor [R68] or [R69]. The placement depends on the platform and is the result of the support of different radio modules. The concerning pin of button 1 is used by the ARM based dresden elektronik radio module to support the Ethernet interface. The pin of button 2 can only be used, if the interrupt feature of the acceleration sensor BMA150 is disabled.

Button 1 [SW1] on deRFnode: Pin 20
Button 1 [SW1] on deRFgateway: Pin 14
Button 2 [SW2]: Pin 29 by setting jumper [JP1] to pin 1 and 2.

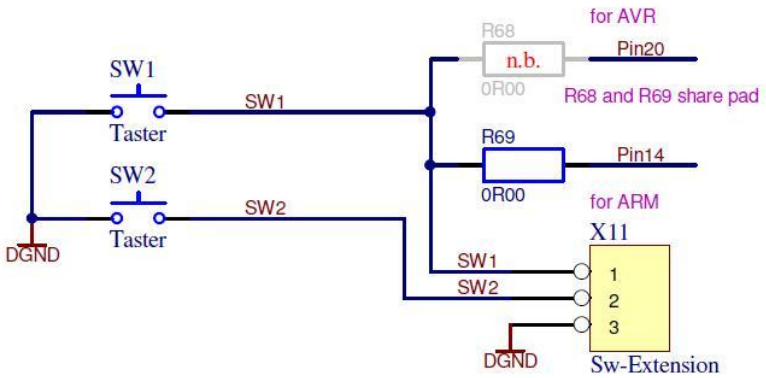


Figure 17: User buttons

8.4. USB interface

Regarding the USB interface, the platforms deRFnode and deRFgateway come in two different variants. The deRFgateway USB interface can only be accessed over native USB of the ARM based dresden elektronik radio modules. The deRFnode platform is offered in two variants. One with native USB for deRFarm7 radio module and another variant with an USB serial converter for deRFmega128 radio modules.

8.4.1. Native USB only for ARM based radio modules

The native USB interface is optimized for using the deRFgateway platforms together with deRFarm7 radio modules, which contain a SAM7X512 microcontroller with an implemented native USB interface. All necessary external parts for USB communication are placed on the deRFgateway.

USBDM: Pin 4
USBDP: Pin 32

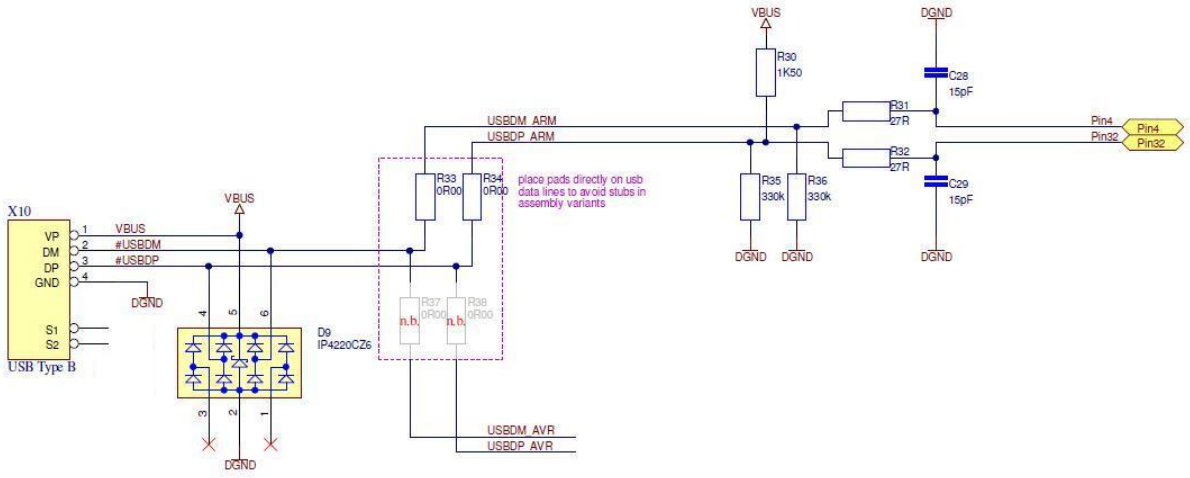


Figure 18: Native USB interface

8.4.2. USB serial for AVR based radio modules

The serial USB interface is a variant of the deRFnode platform. The communication is realized by the transceiver circuit FT245RL **[6]** and the level shifter circuits TXB0108 **[7]**.

Following USB data lines are used:

USB Bit 0: Pin 16
USB Bit 1: Pin 41
USB Bit 2: Pin 12
USB Bit 3: Pin 6
USB Bit 4: Pin 37
USB Bit 5: Pin 19
USB Bit 6: Pin 39
USB Bit 7: Pin 18

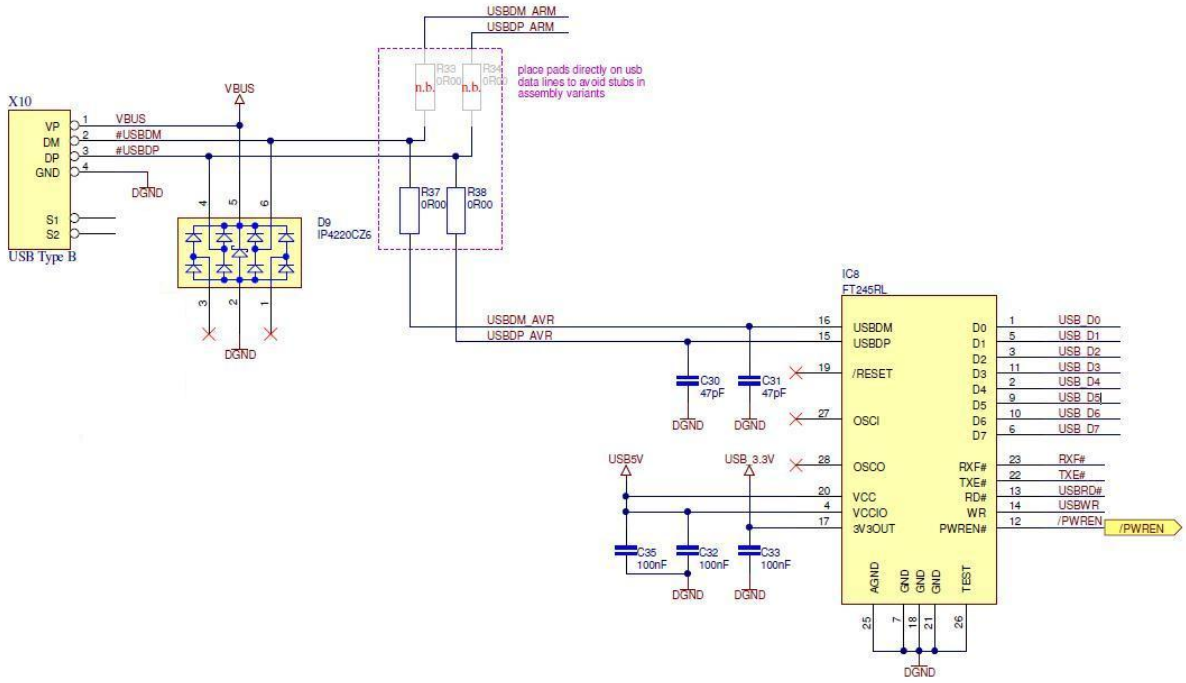


Figure 19: Serial USB interface part 1

The control of the FT245RL is established over the following signals:

USBRD: Pin 27
USBWR: Pin 10
RX: Pin 30
TX: Pin 21

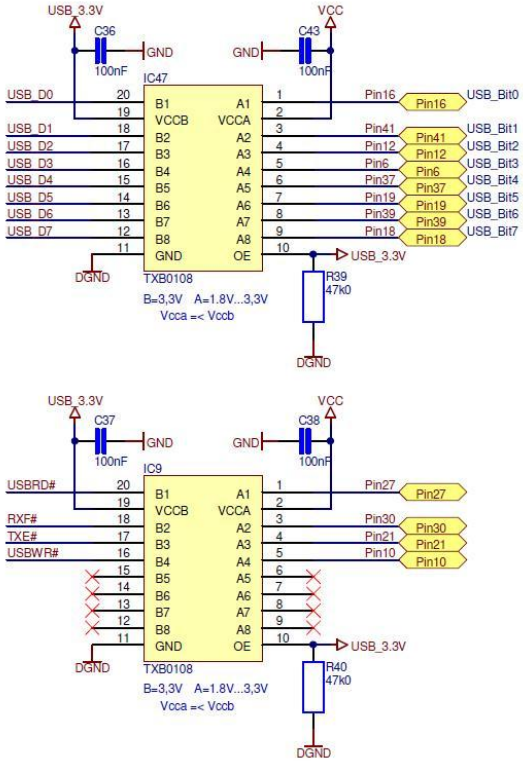


Figure 20: Serial USB interface part 2

8.5. Ethernet

The Ethernet interface is offered only on the deRFgateway platform and can be used in combination with the deRFarm7 radio module, which contains an Ethernet-MAC.
The deRFgateway is assembled with the Ethernet-PHY DP83848I **[8]** and runs in RMII-mode with a 50 MHz clock.

The Ethernet-PHY is connected with the radio module microcontroller over the following signal pins:

ETX0:	Pin 12	ERXER:	Pin 39
ETX1:	Pin 6	ERXDV:	Pin 21
ERX0:	Pin 37	EMDC:	Pin 18
ERX1:	Pin 19	EMDIO:	Pin 30
		MDINTR:	Pin 20

ETXEN: Pin 41

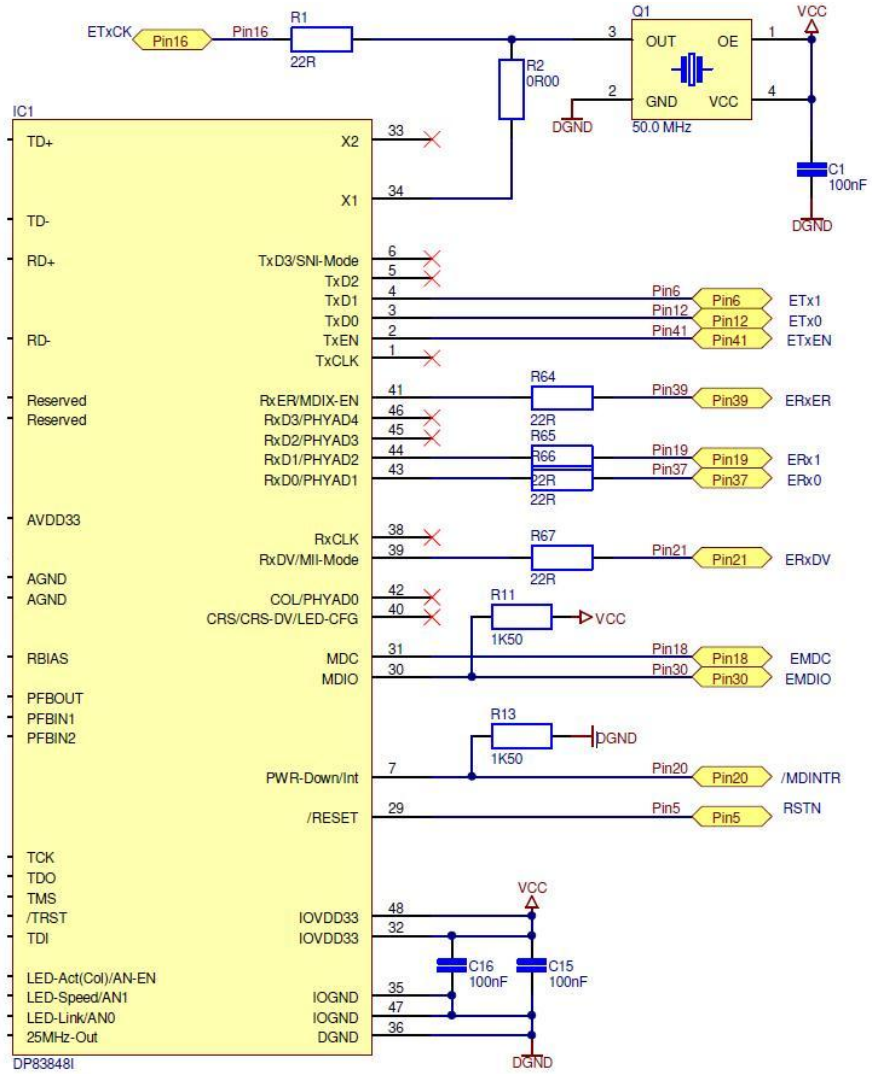


Figure 21: Ethernet-PHY DP83848I

8.6. Power supply

The deRFgateway and deRFnode platform have two different low-dropout regulators (LDO). The first variant is a fixed 3.3VDC supply voltage output for DC and/or USB-powered applications like deRFgateway with Ethernet. The second variant is an adjustable supply voltage output, a low quiescent current LDO, the output voltage can be configured by assembling the resistors [R56] and [R57]. Details can be found in the respective datasheets **[9]**, **[10]**.

Fixed LDO: TPS79433DGN
Adjustable LDO: TPS78001

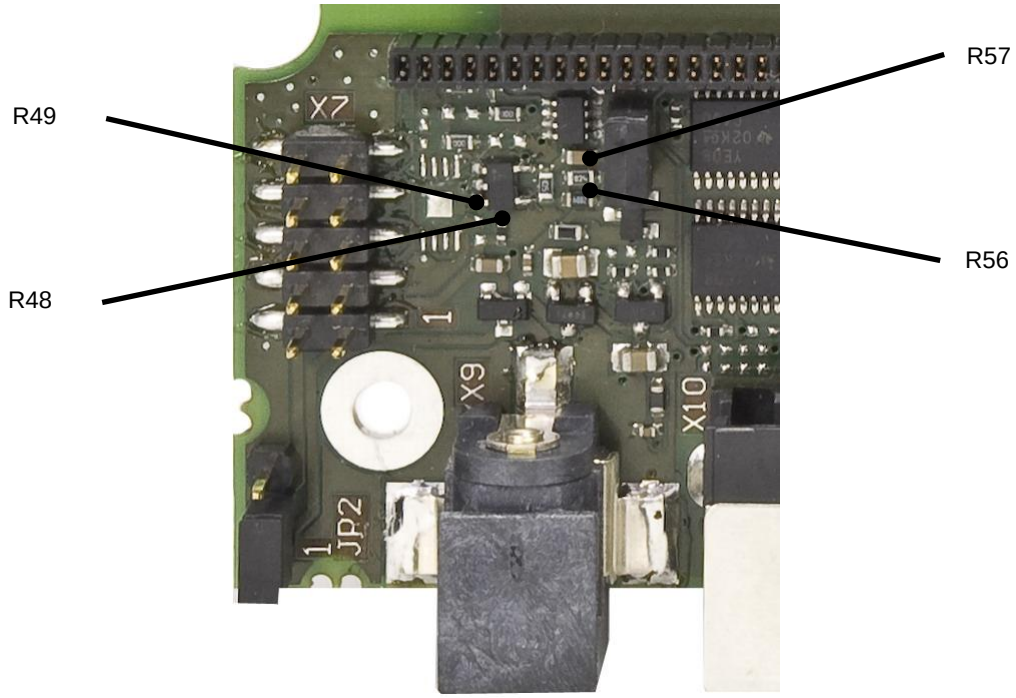


Figure 22: Resistors for adjustable supply voltage

Table 21: LDO configuration

Power supply LDO	**LDO**		**Configuration**				
Platform	*fixed*	*adjustable*	*[R48]*	*[R49]*	*[R56]*	*[R57]*	*Vout*
deRFnode-1TN2P-00N00	**x**		n.a.	0R	n.a.	n.a.	3.3VDC
deRFnode-2TN2P-00N00		**x**	0R	n.a.	2M	820K	3.3VDC
deRFgateway-1TN2P-00N00	**x**		n.a.	0R	n.a.	n.a.	3.3VDC

Table 22: Adjustable LDO configuration

Power supply LDO	Adjustable voltage configuration					
	default	*[R48]*	*[R49]*	*[R56]*	*[R57]*	*Vout*
TPS78001	**x**	0R	n.a.	2M	820K	3.3VDC
		0R	n.a.	n.a.	820K	2.7VDC
		0R	n.a.	2M	n.a.	1.8VDC

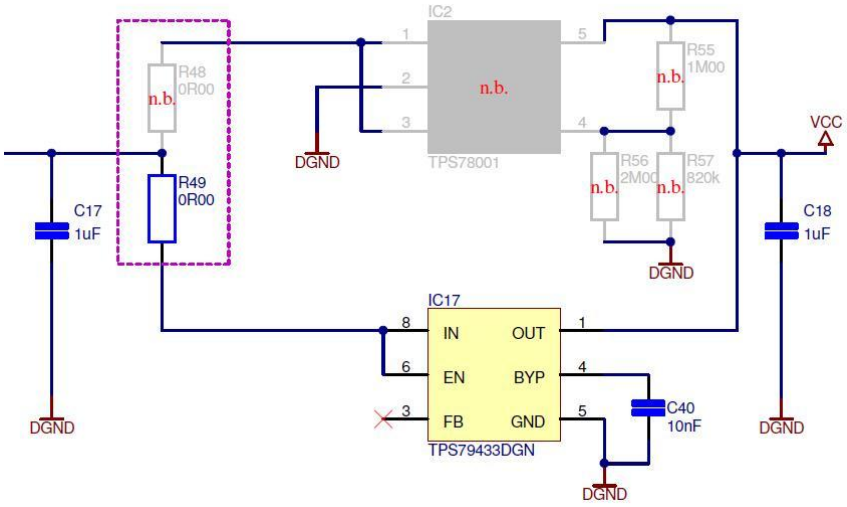


Figure 23: Fixed LDO TPS79433DGN

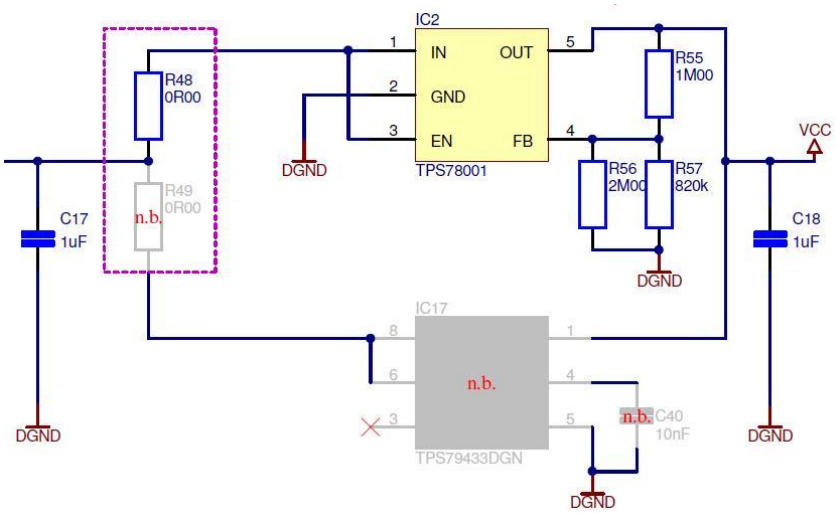


Figure 24: Adjustable LDO TPS78001

8.7. Supervisor

The assembled low-power supervisor LTC2935ITS8-1 **[11]**has selectable threshold voltages. They can be set by the 0R resistors [R58], [R59], [R60], [R61], [R62] and [R63]. If the voltage level drops below the threshold, the supervisor sets a low active reset on Pin 5. This should provide an optimal function of deRFnode and deRFgateway. However in some cases it is not desired to get such a reset. If the jumper [JP4] is removed, the supervisor reset signal will not affect the circuit.

Reset: Pin 5

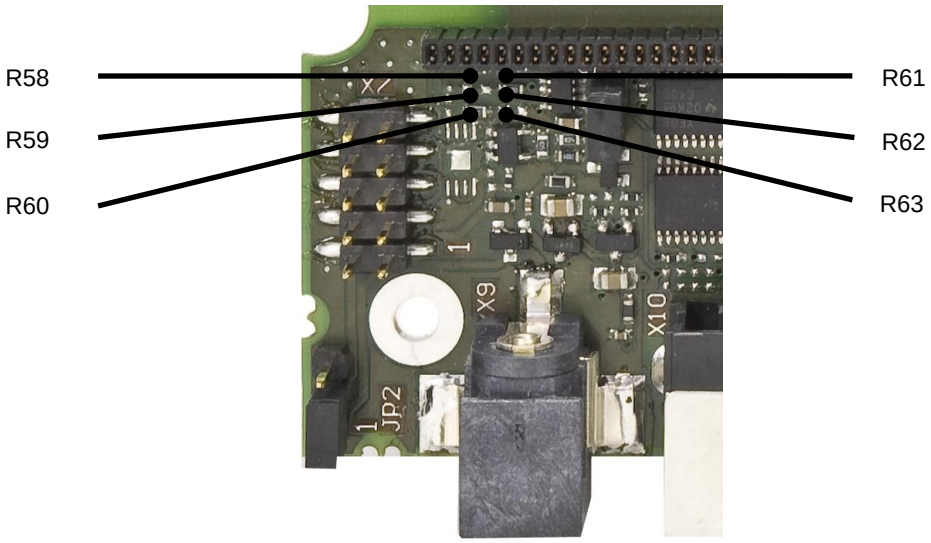


Figure 25: Resistors for supervisor configuration

Table 23: Supervisor configuration

Supervisor							
Platform	*[R58]*	*[R59]*	*[R60]*	*[R61]*	*[R62]*	*[R63]*	*Threshold*
deRFnode-1TN2P-00N00	n.a.	0R	0R	0R	n.a.	n.a.	3.0 VDC
deRFnode-2TN2P-00N00	0R	n.a.	0R	n.a.	0R	n.a.	2.4 VDC
deRFgateway-1TN2P-00N00	n.a.	0R	0R	0R	n.a.	n.a.	3.0 VDC

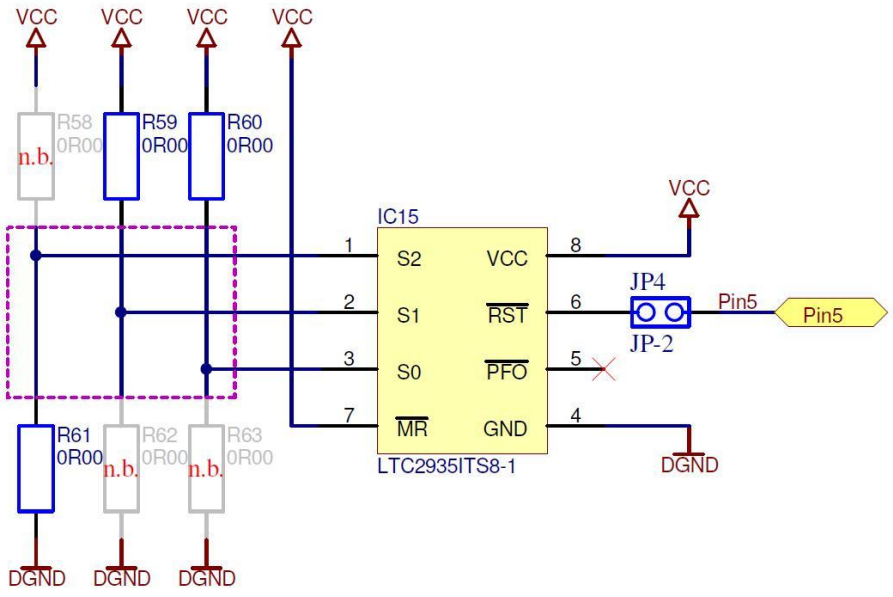


Figure 26: Supervisor LTC2935ITS8-1 with configuration resistors for deRFgateway

8.8. Current measurement

For current consumption tests of the radio module it is possible to place an ampere-meter on jumper [JP5].

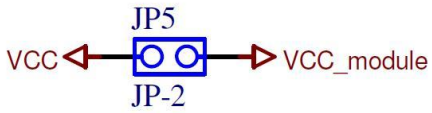


Figure 27: Current measurement for radio module

8.9. USB supply voltage monitoring

The monitoring of the USB power supply can be used as USB detection. The USB voltage can be detected over a voltage divider on Pin 36.

USB detection: Pin 36

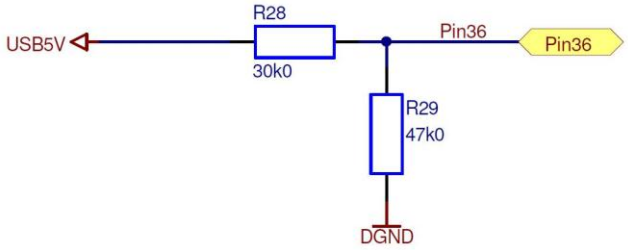


Figure 28: USB supply voltage monitoring

8.10. Battery supply voltage monitoring

It is useful to monitor the supply voltage of battery powered devices to detect low battery charge. The disadvantage is a quiescent current because of the assembled voltage divider. The battery monitoring can be activated by setting jumper [JP3] and read from Pin 34.

Battery monitoring: Pin 34

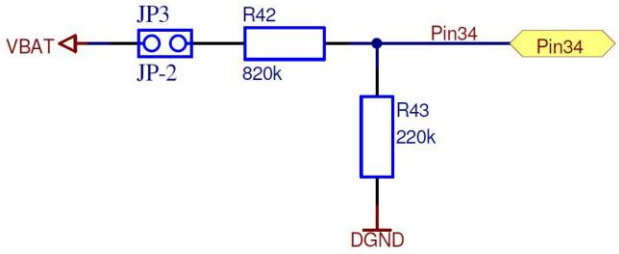


Figure 29: Battery supply voltage monitoring

9. Case variants

For some applications it is useful to cover the PCBA with a suitable case, like fixing deRFnodes and deRFgateways on the wall within a building. Dresden elektronik offers different case variants, depending on board features:

- deRFcase Node
- deRFcase Gateway
- deRFcase Node with external SMA female antenna port
- deRFcase Gateway with external SMA female antenna port

The case will be delivered in a pre-installed state:

- 2x external buttons
- 4x lightpipes for LED and ambient light sensor
- Socket for deRFnode or deRFgateway
- Mounting material like screws and shims
- Fixed pigtail with SMA connector for antenna version

The case has a size of 81 x 85 x 40 mm (L x W x H) with a light grey color and a protection rating of IP40. Please refer to Install Instructions for deRFcase on dresden elektronik homepage.

Figure 30: deRFnode case

Figure 31: deRFgateway case

Figure 32: deRFnode case with antenna connector

Figure 33: deRFgateway case with antenna connector

10. Programming

10.1. Requirements (HW/SW)

Usually deRF-radio modules are shipped with a firmware not meeting your custom application requirements. Exchanging the modules firmware requires:

(1) a firmware binary file,
(2) a suitable JTAG programming adapter,
(3) some programming software.

The following chapters give an overview of creating your own firmware and downloading it onto your target. Generally we suggest using MS Windows® as your developing platform, other operating systems are not supported yet. Depending on the radio module used, additional programming hardware may be required.

10.2. Source code and compiler toolchain

If you bought one of our deRF-development kits, the included CD-ROM contains ready-to-use application-example firmware binary files. In any other case or if you like to build custom firmware, source code files and a suitable compiler toolchain are required additionally. A good starting point for developing wireless software are the “Atmel IEEE 802.15.4 MAC Software Package” **[13]** or the “Atmel bitcloud“ **[14]**. Our Kit-CD already provides the platform adaptions necessary to operate these protocol stacks with your target hardware.

As compiler toolchain, we suggest to use gcc. When working with an AVR MCU, the versions needed are avr-gcc respective arm-none-eabi-gcc for ARM-based MCUs. Basically the IAR™ compiler may also be used but is not supported by dresden elektronik. Avr-gcc is included within the winavr package **[15]**, arm-none-eabi-gcc comes with the yagarto-compiler package **[16]** and is extended by the yagarto tools **[17]** Avr-gcc is also part of Atmel Studio **[18]**.

Supported versions of both are available on our Kit-CD. When downloading, explicitly pay attention to use the version stated since newer releases may cause problems and are unsupported.

10.3. Programming and adapter selection

Please refer to our User Manual “Software Programming“ **[19]** regarding suitable programmers and the programming procedure.

10.4. Software programming model

This chapter is designated to describe how selected components of the deRFnode/gateway-baseboard series may be accessed from a developer’s point of view. As stated in the previous chapters, we only support the GCC, so the following explanations may not fully work with different compilers. Since this should not become a programming tutorial, please refer to the individual device datasheets for details. Also the code snippets given in the following chapters are only extracts. If you like to use them, it’s up to you, to surround them with a working main application and add inclusions for necessary header files that may have been left.

For ARM-MCU based development, the AT91SAM7-Software Package **[20]** delivers, besides many application examples, the so called AT91Library, a large collection of higher-level-functions simplifying the development process. Another starting point might be the Atmel Software Framework (ASF) which comes with Atmel Studio. However ASF is not considered within this chapter. All following chapters base on the AT91SAM7-Software Package. Download and extract it to a directory of your desire. Now change to the ‘packages’ subdirectory, where you will find a set of application examples, one archive per application and compiler.

Delete all non-GNU-based examples and extract the remaining to a common directory, confirm to overwrite files when asked. This is necessary, since the AT91Lib is provided in parts only where each is specially tailored to the belonging application example. Any path stated in future is to be seen relative to `<extraction_root_directory>\at91sam7x-ek\packages\<common_example_directory>\at91lib`.

10.4.1. Enabling the reset supervisor

As already described in chapter 3 the reset supervisors output pin is routed to the MCUs reset pin if JP 4 is shortcut. When using an AVR MCU, no further configuration is required. But when using an ARM MCU, the reset pin functionality must be explicitly enabled. The required library module is `/peripherals/rstc/rstc.c`; an invocation of `RSTC_SetUserResetEnable()` may be used to enable or disable it.

10.4.2. Initialize and use I²C devices

All of the deRFnode/gateway boards include three environmental sensors which are accessible via the two-wire (aka I²C)-Interface (TWI). To use them, you basically have to perform the following steps:

(1) enable the TWI bus
(2) make the MCU the TWI master,
(3) configure the devices behavior and
(4) communicate with the device.

The first step is only necessary if you use deRFmega128-based radio modules. Here you explicitly have to enable the both pull-up's on the SDA/SCL lines by setting PD6 to Low level:

```
DDRD  |=  (1<<PD6);                // make PD6 an output pin
PORTD &= ~(1<<PD6);                // switch to Low level
```

The second step is quite simple since there already exist ready-to-use functions. On AVR MCUs, you may use Peter Fleury's library or its improved version from Manfred Langemann (ask your favourite search engine). On ARM, the AT91Lib provides equivalent functionality under `drivers/twi` and `peripherals/twi`. Independently of the implementation, you usually must decide for a interface speed. 100 up to 400kHz is a good value. TWI libraries may either run interrupt-driven or in polling mode. The latter case is sufficient for first tests while if using the I²C interface excessively, interrupt-based implementations should be preferred.

Assumed that you use an ARM MCU, the initialization might look like:

```
#define PINS_TWI           { ((1<<10) | (1<<11)), AT91C_BASE_PIOA,
                             AT91C_ID_PIOA, PIO_PERIPH_A, PIO_DEFAULT }
static Twid twid;                          // managing datastructure
const Pin pins[] = { PINS_TWI };           // SDA/SCL pins (PA10, PA11)
PIO_Configure(pins, PIO_LISTSIZE(pins));
PMC_EnablePeripheral( AT91C_ID_TWI);       // enable twi peripheral
TWI_ConfigureMaster(AT91C_BASE_TWI, 100000, BOARD_MCK); // 100kHz
TWID_Initialize(&twid, AT91C_BASE_TWI);    // initialize datastructure
AIC_ConfigureIT(AT91C_ID_TWI, 0, ISR_Twi); // configure and
AIC_EnableIT(AT91C_ID_TWI);                // enable twi interrupt
```

During the next step the devices are configured. This includes activity intervals, resolution/sensitivity, triggers when exceeding/falling below given limits, etc. Usually the sensors power up idle and must be explicitly started. Additionally the acceleration sensor includes a configuration EEPROM in which an overriding startup-configuration may be saved. Configuration is usually done by writing to device registers and incorporates:

(1) initiate a TWI start condition,

(2) write the configuration register address,
(3) write the configuration register value,
(4) send a TWI stop condition.

So if you i.e. want to activate the TMP102 temperature sensor measuring temperatures only upon request, select the configuration register MSB (`0x01`) and write in `0x80` to shut down the device. Here `TWID_Write()` encapsulates all the required steps in one function:

```
#define BOARD_SENS_ADDR_TEMP     (0x48)  // sensor address, 1bit shifted
unsigned char ucBuf[2];                  // buffer for twi transmissions
ucBuf[0] = 0x01;                         // configuration register, MSB
ucBuf[1] = 0x80 ;                         // shutdown-mode
TWID_Write(&twid, BOARD_SENS_ADDR_TEMP, 0x00, 0x00, ucBuf, 0x02, NULL);
```

As like as configuration is performed, sensor values are read from device registers. Depending on the device, you may either read the current register value directly or must send a start command first and wait a certain time until measurement is available (otherwise you would read outdated values). To continue with the temperature sensor, a code snippet looks like:

```
ucBuf[0] = 0x01;                         // configuration register, MSB
ucBuf[1] = 0x81;                         // shutdown mode | one-shot
TWID_Write(&twid, BOARD_SENS_ADDR_TEMP, 0x00, 0x00, ucBuf, 0x02, NULL);

// wait at least 26 ms (depends on selected resolution), then
// select temperature register (MSB) and read 2 bytes from it
ucBuf[0] = 0x00;
TWID_Write(&twid, BOARD_SENS_ADDR_TEMP, 0x00, 0x00, ucBuf, 0x01, NULL);
TWID_Read(&twid, BOARD_SENS_ADDR_TEMP, 0x00, 0x00, ucBuf, 0x02, NULL);

// convert value to a human-readable format ...
```

Besides the I^2C communication lines, the acceleration sensor includes an interrupt line which may trigger under certain circumstances, i.e. acceleration increases above/decreases below/changes relatively to a configured threshold. These features might be used to detect the device falling or its motion at all. For all these cases, the sensor might drive its INT line high, as long as the condition is met. For detailed information, please refer to the BMA150 datasheet. Using this feature requires JP1 to shortcut pins 2-3 and configuration of an interrupt trigger on the MCU side.

If you own a dresden elektronik deRFdevelopment-Kit, the included kit CD provides a complete I^2C-library as well as out-of-the box working application examples for AVR and ARM which may be easily modified according to your needs.

10.4.3. Using the USB interface

The onboard USB interface is realized either native (on deRFnode/gateway for ARM) or based on a FTDI USB to parallel FIFO (on deRFnode/gateway for AVR). In USB-speech, ‘native’ means the MCU is able to talk directly to the USB-DM-/-DP-lines which is true for ARM MCUs, but not for AVRs. So if using an AVR MCU on a ARM-baseboard, USB is non-functional. When incorporating the USB FIFO, 12 GPIO (8 data, 4 control) lines are reserved, AND-/OR-gates and an octal driver equalize level differences.

Besides other applications of the USB interface, we focus only on a Communication Device Class (CDC)-device here which simplified is a RS232-port tunneled over USB providing a virtual COM-Port on the PC-side.

For native USB with ARM MCUs, again the AT91Lib provides ready-to-use functions which are to be found in `usb/device/core/USBD_UDP.c` and `usb/device/cdc-serial/CDCDSerial-Driver.c`. A typical initialization looks like:

```
CDCDSerialDriver_Initialize();  // initializes the CDC driver
```

```
USBD_Connect();                     // connects external Pullup to USBDP
```

Afterwards data could be shared with the device driver with `CDCDSerialDriver_Read()` and `CDCDSerialDriver_Write()`, i.e. in your applications main loop.

When working with FTDI, the chip encapsulates the USB-protocol. The only interface are the control (RD, WR, RXF, TXE)- and data lines. Here the initialization procedure consists of:

(1) switching RD and WR to output pins, internal pull-up's enabled,
(2) set RXF, TXE and data lines as input pins,
(3) optionally empty USB FIFO buffer by performing dummy reads.

Data bytes available for reading are signalized by RXF being L as long as the buffer is not completely empty. A byte can be read by toggling RD from H to L, then get the state on all 8 data lines and put RD back to H level.

Writing bytes is performed vice versa: First check the level on TXE - If it is H, the transceiver is busy or the internal buffer is full. Otherwise the transmission may start by setting the 8 data lines according to the transmit byte, then toggle WR from H to L and back. To improve performance, writing should always be done block wise.

The octal bus driver is automatically enabled by the internal USB_CE-signal if at least one of the RD- or WR- goes L (means reception/ transmission in progress). On the other side you should ensure that RD and WR are permanently driven H if you do not use the USB over FDTI to prevent the bus driver energizing back the FTDI circuit.

For simplified detecting whether a USB cable is plugged or not, the VBUS signal may be used (active high). On the FTDI variant this only works reliable if RD and WR are driven H.

The device driver required on the PC side can be downloaded from the dresden elektronik website **[21]**. If you bought a dresden elektronik deRFdevelopment-Kit, the included kit CD also provides the drivers and complete application examples which may be easily modified according to your needs.

10.4.4. Measuring the battery voltage

The VBAT signal may be used to monitor the current battery voltage using the MCUs internal A/D-converter. This requires a reference voltage. On ARM MCUs it must be provided externally on Pin3 which is available on the X8-Header at Pin 32. It is suggested to shortcut it to VCC which is available on the same Header on Pins 2 and 33. Basically AVR MCUs can handle external reference voltages too but we recommend using the internal reference voltage since it avoids additional external shortcut connections and enables a more precise measurement. However in both cases during measurement, the GPIO-Pin 34 (e.g. on X8-Header Pin 15) must not be used otherwise.

The measurement process includes:

(1) Initialization of the ADC,
(2) activation of the required ADC channel (0),
(3) perform the measurement,
(4) shut down the ADC.

For ARM MCUs, in `peripherals/adc/adc.c` the required library functions are to be found and their invocation sequence may look like:

```
#define BOARD_ADC_FREQ 300000         // ADC Frequency
#define ADC_STARTUP_TIME_MAX    20    // returning from Idle mode (µs)
#define ADC_TRACK_HOLD_TIME_MIN 600   // Track&hold Acquisition Time (ns)

unsigned int adc_out, V_bat;
const Pin pin = {1<<27, AT91C_BASE_PIOB, AT91C_ID_PIOB, PIO_INPUT,
                        PIO_DEFAULT};
```

```
PIO_Configure(pin, 1);

ADC_Initialize( AT91C_BASE_ADC,
                AT91C_ID_ADC,
                AT91C_ADC_TRGEN_DIS,
                0,
                AT91C_ADC_SLEEP_NORMAL_MODE,
                AT91C_ADC_LOWRES_10_BIT,
                BOARD_MCK,
                BOARD_ADC_FREQ,
                ADC_STARTUP_TIME_MAX,
                2*ADC_TRACK_HOLD_TIME_MIN);

ADC_EnableChannel(AT91C_BASE_ADC, ADC_CHANNEL_0);
ADC_StartConversion(AT91C_BASE_ADC);

// wait for conversion termination
while (!ADC_IsChannelInterruptStatusSet( ADC_GetStatus(AT91C_BASE_ADC),
                                         ADC_CHANNEL_0)) ;
// perform measurement
adc_out = ADC_GetConvertedData(AT91C_BASE_ADC, ADC_CHANNEL_0);

// convert the measured value to real voltage (mV)
// voltage divider <-> V_bat = 4.73*V_meas
// max. resolution 10bit, V_ref=3.3V <-> V_ref/0x3FF = V_meas/adc_out
// <-> V_bat  = 15.24*adc_out ~ (61*V_meas)/4
V_bat = (61*meas)/4;

// deactivate the ADC for power saving
ADC_DisableChannel(AT91C_BASE_ADC, ADC_CHANNEL_0);
```

For AVR MCUs, an equivalent code snippet is:

```
#define VREF 1.6      // reference voltage; either 1.6 or 3.3 [V]

uint16_t adc_val;     // ADC measurement value
double v_bat;         // real battery voltage
double c ;            // conversion factor

c = ((double)(1040/(double)220)) * VREF ;

#if (VREF==1.6)
    // Select internal 1.6V reference voltage, left AVDREF pin open
    ADMUX = (1 << REFS1) | (1 << REFS0) ;
#else
     // External reference voltage on AVDREF pin, selected by default
#endif

// Analog channel (0) and gain selection (none) <-> MUX5:0 = 0b00000
// -> no changes required

// select prescaler for 500 kHz frequency
#if (F_CPU == (16000000UL))
      ADCSRA |= (1 << ADPS2) | (1 <<ADPS0);    // Prescaler = 32
#elif (F_CPU == (8000000UL))
      ADCSRA |= (1 << ADPS2);                  // Prescaler = 16
#elif (F_CPU == (4000000UL))
      ADCSRA |= (1 << ADPS1) | (1 << ADPS0);   // Prescaler = 8
#elif (F_CPU == (1000000UL))
      ADCSRA |= (1 << ADPS0);                  // Prescaler = 2
#else
```

```
        #error "unsupported F_CPU"
#endif

// put into free running mode (ADTS2:0 = 0b000) -> no changes required

// enable ADC
ADCSRA |= (1 << ADEN);

// Start Conversion, Clear ADIF
ADCSRA |= (1 << ADSC);

// wait for completion
while (!(ADCSRA & (1 << ADIF))) ;

// get measurement
adc_val = ADC;

// disable ADC
ADCSRA &= ~(1 << ADEN);

// convert to real battery voltage (mv)
v_bat = c * adc_val ;
```

10.4.5. Accessing the external flash

Each deRFnode/gateway board provides a serial flash device which is accessed via the SPI interface. Although being from Atmel's AT25 family, if behaves similar to the well-known AT26-Flash devices. So when using an ARM MCU, the AT91 Library functions from `memories/spi-flash/spid.c` and `~/at26.c` may be employed, to allow chip identification you still may add its JEDEC-ID (`0x0001441F`) to the table of device identifiers:

```
/// Array of recognized serial firmware dataflash chips.
Static const At26Desc at26Devices[] = {
   ...
  // Other
  {"AT25DF041"  , 0x0001441F, 1 *  512 * 1024, 256,  64 * 1024}
};
```

A typical initialization of the SPI flash driver may look like:

```
/// SPI0 pin definitions
#define PIN_SPI0_MISO   {1 << 16, AT91C_BASE_PIOA, AT91C_ID_PIOA,
                         PIO_PERIPH_A, PIO_PULLUP}
#define PIN_SPI0_MOSI   {1 << 17, AT91C_BASE_PIOA, AT91C_ID_PIOA,
                         PIO_PERIPH_A, PIO_DEFAULT}
#define PIN_SPI0_SPCK   {1 << 18, AT91C_BASE_PIOA, AT91C_ID_PIOA,
                         PIO_PERIPH_A, PIO_DEFAULT}
#define PIN_SPI0_NPCS0  {1 << 14, AT91C_BASE_PIOA, AT91C_ID_PIOA,
                         PIO_PERIPH_A, PIO_DEFAULT}
#define PINS_SPI0       PIN_SPI0_MISO, PIN_SPI0_MOSI, PIN_SPI0_SPCK

// Base address of SPI peripheral connected to the serialflash.
#define BOARD_AT25_SPI_BASE              AT91C_BASE_SPI0
// Identifier of SPI peripheral connected to the serialflash.
#define BOARD_AT25_SPI_ID                AT91C_ID_SPI0
// Pins of the SPI peripheral connected to the serialflash.
#define BOARD_AT25_SPI_PINS              PINS_SPI0, PIN_SPI0_NPCS0

static Spid spid;       /// SPI driver instance.
Static At26 at26;       /// Serial flash driver instance.
Static Pin pins[] = { BOARD_AT25_SPI_PINS };
```

```
PIO_Configure(pins, PIO_LISTSIZE(pins));
AIC_ConfigureIT(BOARD_AT25_SPI_ID, 0, ISR_Spi);
SPID_Configure(&spid, BOARD_AT25_SPI_BASE, BOARD_AT25_SPI_ID);
AT26_Configure(&at26, &spid, BOARD_AT25_NPCS);
AIC_EnableIT(BOARD_AT25_SPI_ID);
```

Afterwards you may evaluate the number of pages and the pagesize since this is important for each further access:

```
unsigned int numPages = AT26_PageNumber(&at26);
unsigned int pageSize = AT26_PageSize(&at26);
```

Since low-level-accessing the flash device is not trivial, the application Example 'basic-serialflash-project' from the AT91-Library already provides a set of functions necessary to perform read and write operations, erase, protect and unprotect the flash memory. For further details, please refer to `basic-serialflash-project/main.c`.

If you like to access the external flash with an AVR MCU, a code snippet suitable for reading the manufacturer and device Ids (see AT25DF041 datasheet, chapter 10) is:

```
uint8_t I, data[4];

/* Set MOSI, SCK and CS output, all others input */
DDRB = (1<<PB2)|(1<<PB1);
DDRE = (1<<PE5

/* Enable the SPI interface, make the MCU SPI master */
SPCR = (1<<SPE) | (1<<MSTR);

/* Select the serial clock SCK to be (FOSC/4) and double it
 * (i.e. if CPU runs at 8MHz, SPI clock will be 4MHz)
 */
SPCR &= ~((1<<SPR0) | (1<<SPR1));
SPSR = (1<<SPI2X);

/* Start SPI transaction by setting CS low */
PORTE &= ~(1<<PE5);

/* Send the command byte ('Read Manufacturer and Device ID') */
SPDR = 0x9F;

/* wait for termination */
while (!(SPSR & (1 << SPIF))) ;

for(i=0; i<4; i++)
{
  /* Do dummy write for initiating SPI read */
  SPDR = SPI_DUMMY_VALUE;

  /* wait for termination */
  while (!(SPSR & (1 << SPIF))) ;

  /* Upload the received byte in the user provided location */
  data[i] = SPDR;
}

/* Stop the SPI transaction by setting CS high */
PORTE |=  (1<<PE5);

/* check the read Ids (must be 0x1F,0x44,0x01,0x00) ... */
```

As like as in all previous chapters, a ready-to-use library is provided on the CD-ROM belonging to the deRFdevelopment Kit.

10.4.6. Initialize and use the Ethernet transceiver

Only the deRFgateway boards are equipped with Ethernet circuitry hardware (PHY layer transceiver). The EMAC must be implemented in software. Although it is not impossible to do this with an AVR, we focus on using it with an ARM MCU here since as like as in all chapters before the AT91Lib already provides this functionality.

Transceiver and MCU are connected via RMII (Reduced Media Independent Interface). To save energy, the Ethernet transceiver starts up in power-down mode (pull-down resistor on transceiver pin 7). Alternatively the same port pin may act as an interrupt line. Due to a pull-down resistor, this is not possible in factory state. If you intend to use it, please remove R13.

The Ethernet initialization procedure consists of:

(1) Setup the EMAC (enable the EMAC peripheral, configure the PIO pins and the clock interface between MCU and transceiver, set the MAC address).
(2) Initialize the PHY (power up, setup up connection preferences like Auto negotiation, LinkSpeed and Duplex behavior). This is done by writing to the PHYs register set.
(3) Initialize local reception and transmission buffers.
(4) Enable receiver and transmitter.

Afterwards it's up to the application, to process incoming frames and handle changes of the link state (Ethernet cable plugged/unplugged, changes of link speed). Since the transceiver only throws interrupts upon state changes, the application has to poll for incoming frames.

Due to complexity, we skipped printing code snippets here. If you bought a deRFdevelopment kit, the included Kit CD provides the respective application source code entirely. Otherwise a good starting point is the At91Lib's `basic-emac-project-application` example. All EMAC Library functions are to be found in `peripherals/emac/emac.c` while the PHY transceiver abstraction resists under `components/ethernet/`. The AT91Lib assumes you have a DM9161 transceiver which unfortunately is not compatible with the DP83848C assembled on the deRFgateway board. For further information, please refer to the datasheets.

10.4.7. Minimize device power consumption

Optimizing the energy consumption is especially important if the device is battery powered to ensure a long battery lifetime. If the USB cable is plugged, the device gets its power through the USB line; in this case it makes no sense to think about power down modes. If finally Ethernet shall be used, the power consumption is too large for reasonably powering the device via batteries, so here DC power should be used which implies that the energetic optimization is also obsolete. Reference values of power consumption are given in **Section 5.3**.

The following list describes what might be done, to decrease the power consumption. Depending on your application requirements, not all points may be realizable.

(1) Power down the Ethernet transceiver:

If you did not explicitly activate it, the transceiver already is powered down. Otherwise configure the transceiver PWR_DOWN/INT line to be a Power-Down line by writing a logical zero to the MII Interrupt control register (MICR, address 0x11) bit 0. Also ensure that the MCU pin connected to the PWR-DOWN-line has no internal pull-up activated. The onboard pull-down resistor will now force an Ethernet transceiver power down.

(2) Power down the I²C sensors:

In factory default state and if you didn't configure them, only the acceleration sensor is active. Although we recommend to explicitly disable all three sensors. This comprises of sending stop conversion commands as well as disabling any auto-conversion mode.

(3) Power down the radio transceiver:

This step depends on the radio module used. If you have a deRFmega128–module, the respective power-reduction register has to be used:

```
#include <power.h>
PRR1 |= (1<<PRTRX24);      // power down transceiver
```

Any other Atmel radio transceiver (e.g. AT86RF212 on deRFarm7 series radio modules) has an internal state machine which may be switched to sleep state by toggling the levels on the SLP_TR and /RST lines:

- SLP_TR L, /RST L → force RESET state
- wait 1µs, then set /RST H → TRX_OFF state
- put SLP_TR H → SLEEP state
- leave SLP_TR H (setting it back to L would result in transition back to TRX_OFF)

The following diagram (taken from AT86RF212s datasheet) illustrates the procedure:

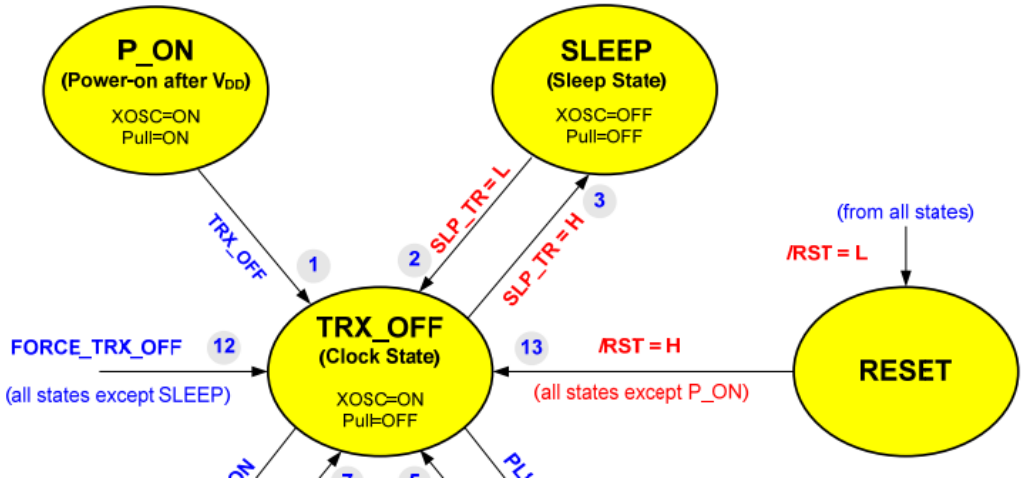


Figure 34: State control (source: ATMEL™ AT86RF212 transceiver datasheet [12])

(4) Remove any unneeded external cabling such as a level shifter or your JTAG programming adapter (if your application allows).

(5) Switch off the onboard LEDs.

(6) Put the onboard Flash to Deep-Power-Down-Mode.

(7) Disable the onboard USB FIFO (only deRFnode/gateway for AVR):

On boards equipped with a FTDI USB FIFO, the USB transceiver may be disabled only by physically disconnecting the USB cable. Due to a hardware issue it may still happen, that the octal driver (IC9) energizes back the FIFO (IC8). To avoid that, ensure, that (in contrast to (8)), RD and WR always drive high current.

(8) Ensure that the MCU GPIO pins do not drive current.

Due to circuit design, some GPIO pins are set to GND, others have external pull-up's or pull downs applied which are required for proper operation, but when sleeping, they still

drive unnecessary current. Regardless of the used MCU, this may be achieved by first setting the pins to be outputs; internal pull-up's enabled. Then read back the level on each pin – if it differs from the expected state (being H due to enabled pull-up's), disable individual pull-up's. Be careful to leave out the Pins connected to the transceiver or the USB FIFO, as otherwise you would reactivate the devices powered down before (see above).

(9) Disable any unnecessary MCU-internal device. This includes running timers, transceivers (UART, DBGU, TWI, SPI, native USB), ADC, watchdog.

On AVR MCUs, this may be achieved by writing to the Power Reduction Registers, like:

```
#include <avr/power.h>
PRR0 = (1<<PRTIM0);     // disable Timer 0
```

or using existing functions from `power.h`, such as `power_timer1_disable()`.

On ARM MCUs the equivalent function call is `PMC_DisablePeripheral(id)` from `peripheral/pmc/pmc.c` where the peripheral Ids are to be found in the At91SAM7X-datasheet. If you desire to switch off all devices at once, `PMC_DisableAllPeripherals()` will do that.

The watchdog is configured slightly different. On AVR MCUs use:

```
#include <avr/wdt.h>
wdt_disable();
```

respective WD_Disable() from peripheral/wdt/wd.c on ARM MCUs.

If using native USB on ARM MCUs, the USB transceiver is disabled by invoking `UDP_DisableTransceiver()` from `usb/device/core/USBD_UDP.c`.

(10) Disable BOD:

Slight improvements may be achieved by disabling the Brown-out-detection feature. On ARM MCUs, the GPNVM Bit0 has to be cleared by writing to the internal flash memory:

```
EFC_PerformCommand(AT91C_BASE_EFC0, AT91C_MC_FCMD_CLR_GP_NVM,
        AT91C_MC_GPNVM0);
```

On AVR MCUs this is done by clearing the BODLEVEL bits (2:0) in the Fuse Low Byte.

(11) Slow down the MCU clock:

This differs depending on the radio module used. When it is deRFmega128 series based, simply put the MCU to sleep mode by writing to the sleep mode control register followed by executing the sleep instruction.

```
#include <avr/io.h>          // MCU register definitions
#include <avr/sleep.h>
SMCR = (1<<SE) | (1<<SM1);   // go into power down mode
sleep_cpu();                 // execute sleep instruction
```

If using an ARM based MCU, the AT91Lib usb-device-massstorage-project example provides functions for switching the MCU main clock back to 32kHz.

For waking up the device again you may e.g. leave a timer running or configure a external interrupt trigger. The steps to be performed after wakeup include simplified the reverse procedure as described above, especially reinitialization of internal devices that have been powered down.

If you bought a deRFdevelopment kit, the Kit-CD includes a deRFnative example which demonstrates low-power modes.

11. Ordering information

The ordering code for deRFnode and deRFgateway are listed in **Table 24**.

Table 24: Ordering information

Type code	Order number
plain variant – no radio module included	
deRFnode-1TNP2-00N00	BN-031632
deRFnode-2TNP2-00N00	BN-031634
deRFgateway-1TNP2-00N00	BN-031633
accessories	
deRFcase node	BN-030043
deRFcase gateway	BN-030046
deRFcase node with external SMA female antenna connector	BN-032770
deRFcase gateway with external SMA female antenna connector	BN-032771

Note: **The deRFcase variants do not contain any platforms or radio modules. These components must be ordered separately!**

12. Revision notes

(1) When using PA modules together with deRFnode-2TNP2-00N00, the reset supervisor might unintentionally reset the module when sending frames at high transmit power. Decrease the transmit power (transceiver register TX_PWR set to a 0xA or higher) or remove JP4 to avoid.

(2) Due to discontinuation of Atmel's Flash series, recent versions of the deRFnode/gateway boards are equipped with a Micron M25P40 dataflash. Both command sets are pretty similar, refer to datasheets for details **[4]**, **[5]** .

References

[1] Datasheet Temperature Sensor TMP102AIDRLT
http://www.ti.com.cn/cn/lit/ds/symlink/tmp102.pdf

[2] Datasheet Ambient Light Sensor ISL29020IROZ-T7
https://www.intersil.com/content/dam/Intersil/documents/isl2/isl29020.pdf

[3] Datasheet Acceleration Sensor BMA150
http://ae-bst.resource.bosch.com/media/products/dokumente/bma150/BST-BMA150-DS000-07.pdf

[4] Datasheet Atmel Flash AT25DF041
http://www.datasheetarchive.com/AT25DF041-datasheet.html

[5] Datasheet Micron Flash M25P40
http://www.micron.com/-/media/Documents/Products/Data%20Sheet/NOR Flash/Serial NOR/M25P/M25P40.pdf

[6] Datasheet USB FIFO FT245RL
http://www.ftdichip.com/Documents/DataSheets/ICs/DS_FT245R.pdf

[7] Datasheet Levelshifter TXB0108
http://www.ti.com/lit/ds/symlink/txb0108.pdf

[8] Datasheet Ethernet PHY DP83848I
http://www.ti.com/lit/ds/symlink/dp83848i.pdf

[9] Datasheet Single channel LDO TPS79433
http://www.ti.com/lit/ds/symlink/tps79433.pdf

[10] Datasheet Single channel LDO TPS78001
http://www.ti.com/lit/ds/symlink/tps78001.pdf

[11] Datasheet Power Supervisor LTC2935
http://cds.linear.com/docs/en/datasheet/2935fa.pdf

[12] Datasheet Atmel AT86RF212 IEEE 802.15.4 transceiver
http://www.atmel.com/Images/doc8168.pdf

[13] Atmel IEEE 802.15.4 MAC Software Package
http://www.atmel.com/tools/IEEE802_15_4MAC.aspx

[14] Atmel BitCloud - ZigBee Pro
http://www.atmel.com/tools/bitcloud-zigbeepro.aspx

[15] winavr, version 20100110
http://sourceforge.net/projects/winavr/files/WinAVR/20100110

[16] yagarto GNU arm toolchain
http://sourceforge.net/projects/yagarto/

[17] yagarto tools
http://www.emb4fun.de/download/arm/yagarto/yagarto-tools-20121018-setup.exe

[18] Atmel Studio
http://www.atmel.com/tools/atmelstudio.aspx

[19] User Manual Software Programming
http://www.dresden-elektronik.de/funktechnik/service/ downloads/documentation/
?eID=dam_frontend_push&docID=1917

[20] AT91SAM7X-512 Software package for IAR 5.2, Keil and GNU, revision 1.5
http://www.atmel.com/dyn/resources/prod_documents/at91sam7x-ek.zip

[21] deRFusb Driver
http://www.dresden-elektronik.de/funktechnik/service/downloads/software/
?eID=dam_frontend_push&docID=2327

dresden elektronik ingenieurtechnik gmbh
Enno-Heidebroek-Straße 12
01237 Dresden
GERMANY

Phone +49 351 - 31850 0
Fax +49 351 - 31850 10
Email wireless@dresden-elektronik.de

Trademarks and acknowledgements

- ZigBee® is a registered trademark of the ZigBee Alliance.
- 802.15.4™ is a trademark of the Institute of Electrical and Electronics Engineers (IEEE).

All trademarks are registered by their respective owners in certain countries only. Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such.

Disclaimer

This note is provided as-is and is subject to change without notice. Except to the extent prohibited by law, dresden elektronik ingenieurtechnik gmbh makes no express or implied warranty of any kind with regard to this guide, and specifically disclaims the implied warranties and conditions of merchantability and fitness for a particular purpose. dresden elektronik ingenieurtechnik gmbh shall not be liable for any errors or incidental or consequential damage in connection with the furnishing, performance or use of this guide.