dresden elektronik

User Manual
deRFnode / deRFgateway

Document Version V1.1
2011-07-15

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 10
   4.1. Battery powered nodes in a small wireless sensor network 10
   4.2. 6LoWPAN tree-network application 10
   4.3. Point-to-Point connection for simple applications 10

5. Technical data 11
   5.1. Mechanical 11
   5.2. Operation conditions 11
   5.3. Electrical 11
      5.3.1. Operational ranges 11
      5.3.2. Current consumption 12

6. Overview of platforms 14

7. Pin assignment 17
   7.1. Radio module interface 17
   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 24

8. Board features 25
   8.1. Onboard sensors 26
      8.1.1. Temperature sensor 26
      8.1.2. Ambient light sensor 26
      8.1.3. Acceleration sensor 27
   8.2. LEDs and buttons 27
      8.2.1. User LEDs 27
      8.2.2. User buttons 28
   8.3. USB interface 28
      8.3.1. Native USB only for ARM based radio modules 29
      8.3.2. USB serial for AVR based radio modules 29
   8.4. Ethernet 31
   8.5. Power supply 32
   8.6. Supervisor 34
   8.7. Current measurement 36
   8.8. USB supply voltage monitoring 36
   8.9. Battery supply voltage monitoring 37

9. Programming 38
9.1. Requirements (HW/SW) 38
9.1.1. Source code and compiler toolchain 38
9.1.2. JTAG programming and adapter selection 38
9.1.3. JTAG programming software 38
9.2. AVR based radio modules over JTAG 39
9.3. ARM based radio modules over JTAG 41
9.4. Software programming model 42
9.4.1. Enabling the reset supervisor 42
9.4.2. Initialize and use I2C devices 42
9.4.3. Using the USB interface 44
9.4.4. Measuring the battery voltage 45
9.4.5. Accessing the external flash 47
9.4.6. Initialize and use the Ethernet transceiver 48
9.4.7. Minimize device power consumption 49

10. Ordering information 52

11. Revision notes 53

Document history

Date	Version	Description
2011-04-15	1.0	Initial version
2011-07-15	1.1	Update feature list

Mailing list

Firm	Division / Name
DE	Dev. / A. Palm

Author / Check / Release

	Firm	Division / Name
Author	DE	Dev. / A. Palm
Check		
Release		

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:

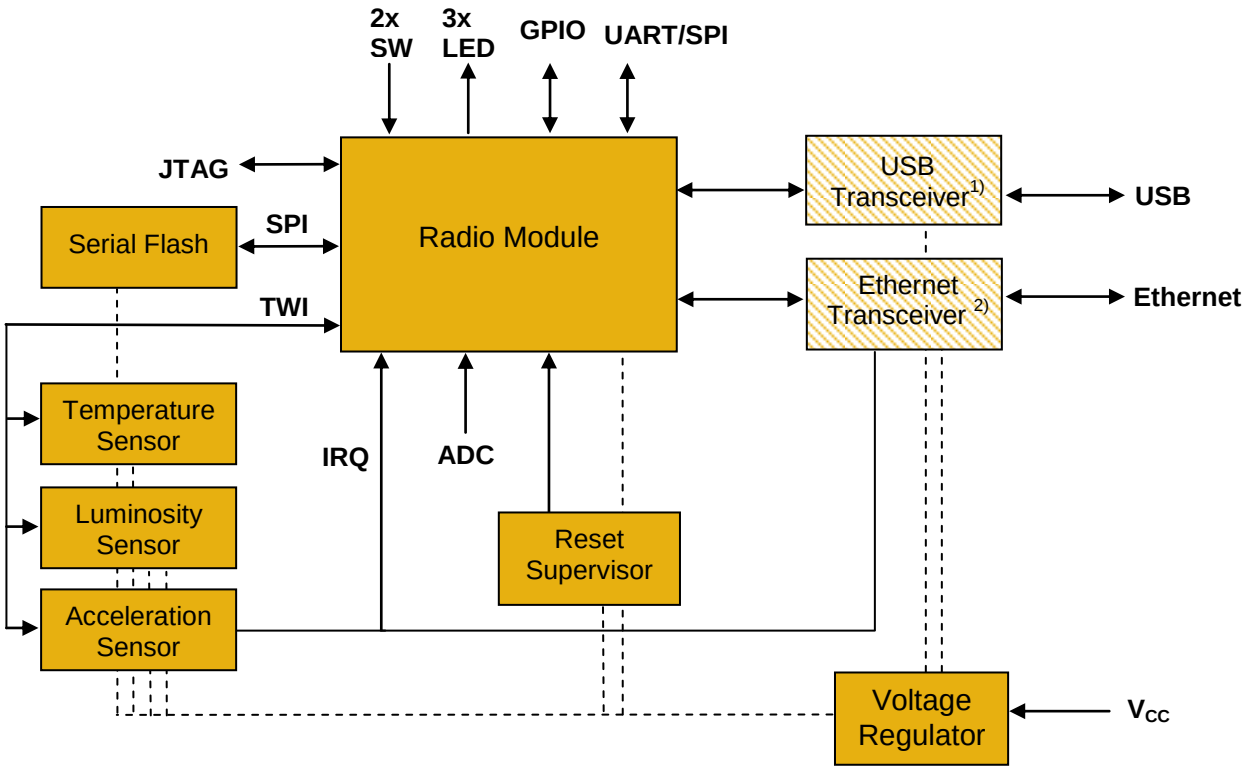
- Stand-Alone End device for a IEEE 802.15.4 compliant network
- applicable as coordinator for small networks
- battery powered applications with a lifetime of several years
- 6loWPAN nodes
- ZigBee®

3. Features

The main features of deRFnode and deRFgateway are:

- Compact size: 69 x 75 x 30 mm
- Supports AVR or ARM based dresden elektronik radio modules
- Variants for pluggable and solderable radio modules
- USB and 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] optionally (only on deRFnode-series with AVR, otherwise included in MCU)

[2] 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.

The portfolio of deRFnodes and deRFgateways will be added with new variants in the future. All available 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
deRFnode-2TNP2-00N00	deRFmega128 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								
			S	SMT								
				A	ARM7X							
				M	ATMEGA							
				N	None (no delivered radio module)							
					P	Plain (only PCBA)						
						2	Revision 2					
radio module properties												
							1	Sub-GHz transeiver				
							2	2.4GHz transceiver				
								2	128k internal Flash			
								5	512k internal Flash			
									A	THT, pluggable		
									C	SMT, solderable		
										0	0	Chip antenna
										0	2	Coaxial plug
other												
							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. The solderable radio modules will not be offered in all variants.

Table 3: Feature list

Platform	Radio module						Supported features						
	deRFmega128-22A00	deRFmega128-22A02	deRFarm7-15A00	deRFarm7-15A02	deRFarm7-25A00	deRFarm7-25A02	Native USB	Serial USB	Ethernet	Sensors	LEDs and Buttons	USB powered	Low Quiescent Curr. LDO
deRFnode													
1TNP2-00N00			x	x	x	x	x			x	x	x	
1TNC2-00N00			x	x	x	x	x			x	x	x	
2TNP2-00N00	x	x						x		x	x	x	x
2TNC2-00N00	x	x						x		x	x	x	x
deRFgateway													
1TNP2-00N00			x	x	x	x	x		x	x	x	x	
1TNC2-00N00			x	x	x	x	x		x	x	x	x	

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. Battery powered nodes in a small wireless sensor network

Application:
A small network consisting of about 30 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.

Required components:

Platforms:	deRFnode-2TNP2-00N00
Radio modules:	deRFmega128-22A00
Software:	based on Atmel's MAC-Stack available on Development-Kit CD

4.2. 6LoWPAN tree-network application

Application:
A wireless network, that can be monitored and controlled via Ethernet by a 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, that 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 and Control Manager on Development-Kit CD

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 available on Development-Kit CD

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 follow:

Supply voltage: VCC = 3.3VDC ± 0.3VDC
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, I2C-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 8.4).

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)]
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	4Mbit serial Flash	2.7V	3.6V	25µA	10mA	20mA
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

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 to the values given above 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

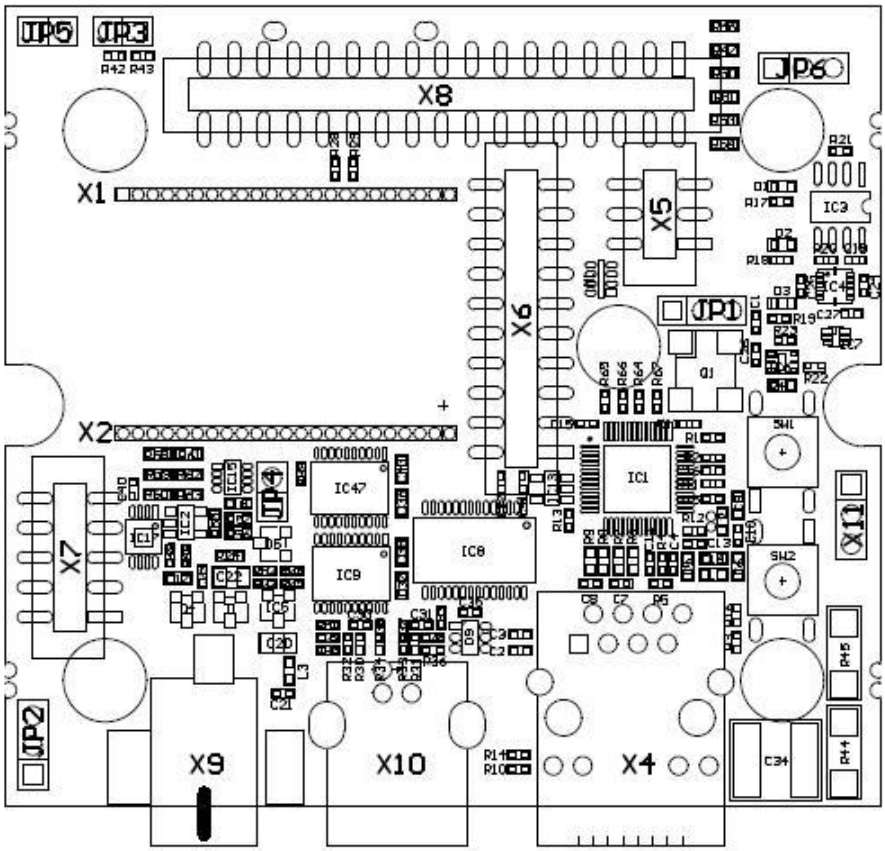


Figure 2: Overlay of deRFnode and deRFgateway

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

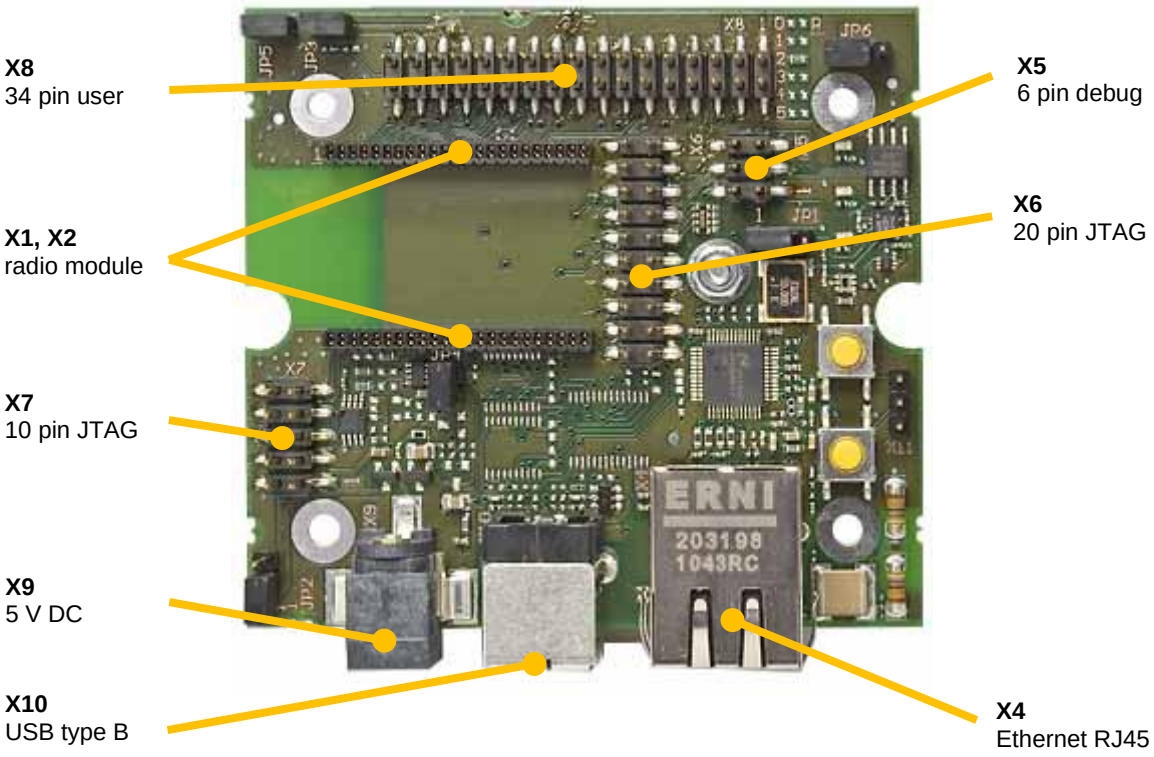


Figure 3: deRFgateway-1TNP2

Figure 4: deRFnode-1TNP2

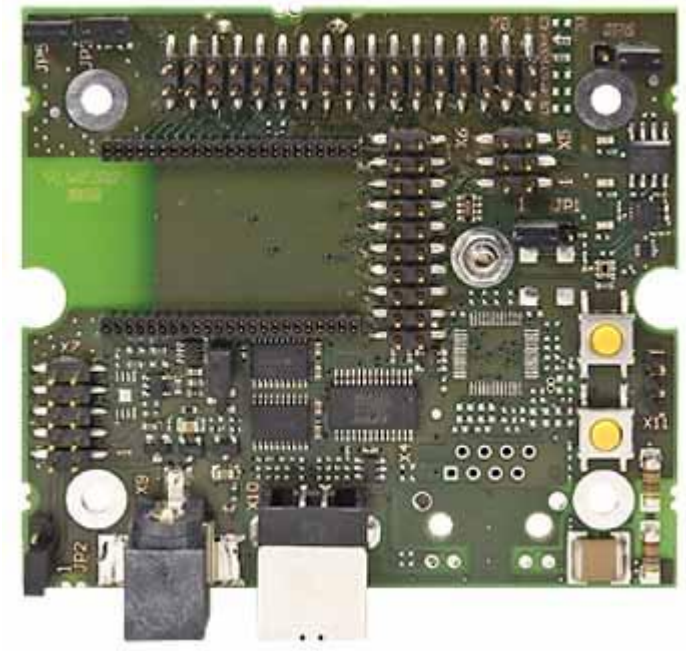


Figure 5: deRFnode-2TNP2

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.

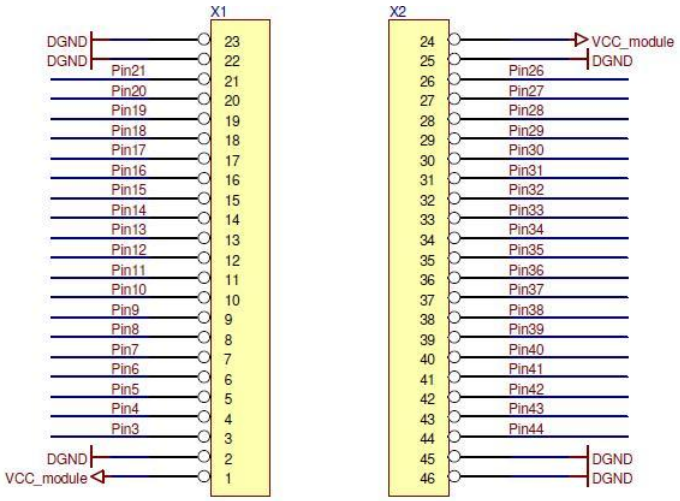


Figure 6: Header for radio modules

The next two tables give an overview of the radio module signals. Table 9 shows the signal names of the deRFmega128 radio module series of dresden elektronik, Table 10 for the deRFarm7 radio module series.

Table 9: Pin assignment for deRFmega128-series 22A00 / 22A02 / 22C00 / 22C02

Pin assignment			
Pin	*µC-Port (deRFmega128)*	*Pin*	*µC-Port (deRFmega128)*
1	VCC	24	VCC
2	GND	25	GND
3	AREF	26	PE0/RXD0/PCINT8
4	PG1/DI1	27	PD2/RXD1/INT2
5	RSTN	28	PE1/TXD0
6	PG2	29	PD6/T1
7	PD0/SCL/INT0	30	PE2/XCK0/AIN0
8	PG5/OC0B	31	PE3/OC3A/AIN1
9	PD1/SDA/INT1	32	PD4/ICP1
10	PD3/TXD1/INT3	33	PE4/OC3B/INT4
11	PD7/T0	34	PF0/ADC0
12	PD5/XCK1	35	PE5/OC3C/INT5
13	PB1/SCK/PCINT1	36	PF1/ADC1
14	CLKI	37	PE6/T3/INT6
15	PB2/MOSI/PCINT2/PDI	38	PF4/ADC4/TCK
16	PB0/SSN/PCINT0	39	PE7/ICP3/CLKO/INT7
17	PB3/MISO/PCINT3/PDO	40	PF5/ADC5/TMS
18	PB6/OC1B/PCINT6	41	PF2/ADC2
19	PB4/OC2/PCINT4	42	PF6/ADC6/TDO
20	PB7/OC0A/OC1C/PCINT7	43	RSTON
21	PB5/OC1A/PCINT5	44	PF7/ADC7/TDI
22	GND	45	GND
23	GND	46	GND

Table 10: Pin assignment for deRFarm7-series 25A00 / 25A02 / 25C00 / 25C02 / 15A00 / 15A02 / 15C00 / 15C02

Pin assignment			
Pin	*µC-Port (deRFarm7)*	*Pin*	*µC-Port (deRFarm7)*
1	VCC	24	VCC
2	GND	25	GND
3	ADVREF	26	PA27/DRXD/PCK3
4	USBDM	27	PA0/RXD0
5	RSTN	28	PA28/DTXD
6	PB3/ETX1	29	PA4/CTS0/SPI1_NPCS3
7	PA11/TWCK	30	PB9/EMDIO
8	PB26/TIOB1/RI1	31	PB21/PWM2/PCK1
9	PA10/TWD	32	USBDP
10	PA1/ TXD0	33	PB19/PWM0/TCLK1
11	PB25/TIOA1/DTR1	34	PB27/TIOA2/PWM0/AD0
12	PB2/ETX0	35	PA14/SPI0_NPCS2/IRQ1
13	PA18/SPI0_SPCK	36	PB28/TIOB2/PWM1/AD1
14	PA3/RTS0/SPI1_NPCS2	37	PB5/ERX0
15	PA17/SPI0_MOSI	38	TCK
16	PB0/ETXCK/EREFCK	39	PB7/ERXER
17	PA16/SPI0_MISO	40	TMS
18	PB8/EMDC	41	PB1/ETXEN
19	PB6/ERX1	42	TDO
20	PB18/EF100/ADTRG	43	JTAGSEL
21	PB15/ERXDV/ECRSDV	44	TDI
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.

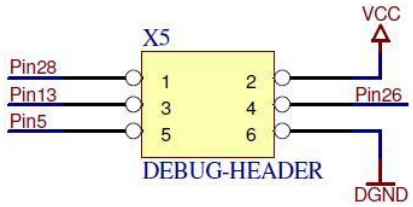


Figure 7: Debug header

The following table shows the signal description.

Table 11: Debug Header Pin assignment

Pin assignment			
Pin	*Function*	*Pin*	*Function*
1	TXD (UART0/DBGU)	2	VCC
3	SCK	4	RXD (UART0/DBGU)
5	RSTN	6	GND

7.3. JTAG for ARM

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

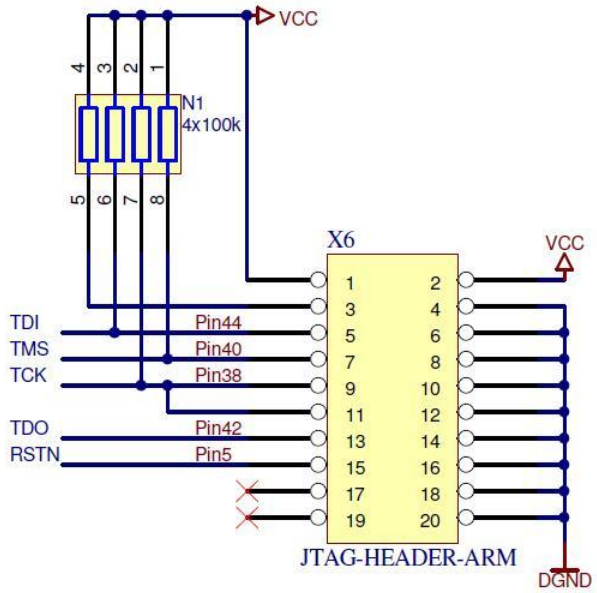


Figure 8: JTAG for ARM header

The following table shows the signal description.

Table 12: JTAG for ARM header pin assignment

Pin assignment			
Pin	*Function*	*Pin*	*Function*
1	VCC	2	VCC
3	100K Pullup	4	GND
5	TDI, 100K Pullup	6	GND
7	TMS, 100K Pullup	8	GND
9	TCK, 100K Pullup	10	GND
11	TCK	12	GND
13	TDO	14	GND
15	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.

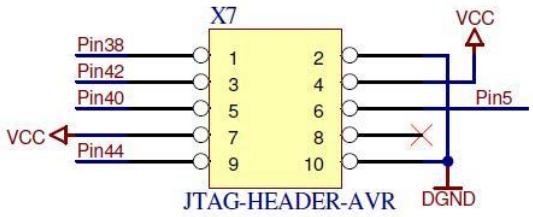


Figure 9: JTAG AVR header

The following table shows the signal description.

Table 13: JTAG for AVR header pin assignment

Pin assignment			
Pin	*Function*	*Pin*	*Function*
1	TCK	2	GND
3	TDO	4	VCC
5	TMS	6	RSTN
7	VCC	8	N/C
9	TDI	10	GND

7.5. User Interface

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

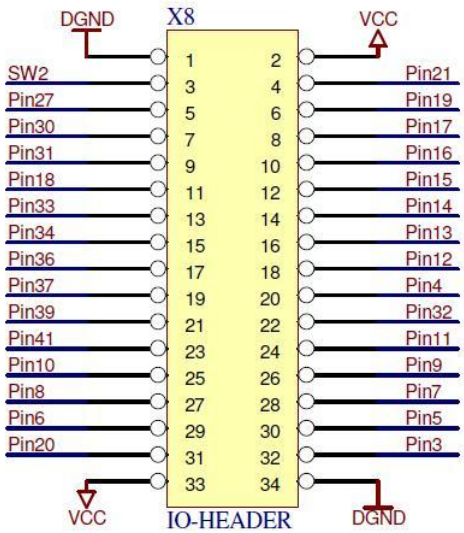


Figure 10: User header

The following table shows the signal description.

Table 14: User interface header pin assignment

Pin assignment		
Pin	*Function deRFmega128*	*Function deRFarm7*
1	GND	GND
2	VCC	VCC
3	SW2 or PD6/T1	SW2 or PA4/CTS0/SPI1_NPCS3
4	PB5/OC1A/PCINT5	PB15/ERXDV/ECRSDV
5	PD2/RXD1/INT2	PA0/RXD0
6	PB4/OC2/PCINT4	PB6/ERX1
7	PE2/XCK0/AIN0	PB9/EMDIO
8	PB3/MISO/PCINT3/PDO	PA16/SPI0_MISO
9	PE3/OC3A/AIN1	PB21/PWM2/PCK1
10	PB0/SSN/PCINT0	PB0/ETXCK/EREFCK
11	PB6/OC1B/PCINT6	PB8/EMDC
12	PB2/MOSI/PCINT2/PDI	PA17/SPI0_MOSI
13	PE4/OC3B/INT4	PB19/PWM0/TCLK1
14	CLKI	PA3/RTS0/SPI1_NPCS2
15	PF0/ADC0	PB27/TIOA2/PWM0/AD0
16	PB1/SCK/PCINT1	PA18/SPI0_SPCK
17	PF1/ADC1	PB28/TIOB2/PWM1/AD1
18	PD5/XCK1	PB2/ETX0
19	PE6/T3/INT6	PB5/ERX0
20	PG1/DIG1	USBDM
21	PE7/ICP3/CLKO/INT7	PB7/ERXER
22	PD4/ICP1	USBDP
23	PF2/ADC2	PB1/ETXEN
24	PD7/T0	PB25/TIOA1/DTR1
25	PD3/TXD1/INT3	PA1/ TXD0
26	PD1/SDA/INT1	PA10/TWD
27	PG5/OC0B	PB26/TIOB1/RI1
28	PD0/SCL/INT0	PA11/TWCK
29	PG2	PB3/ETX1
30	RSTN	RSTN

31	PB7/OC0A/OC1C/PCINT7	PB18/EF100/ADTRG
32	AREF	ADVREF
33	VCC	VCC
34	GND	GND

7.6. Jumper configuration

The following table shows the possible jumper configuration.

Table 15: Jumper configuration

Pin assignment	
JP	*Function*
1	GPIO Input diversity (SW2 = closed pin 1:2 / acceleration sensor interrupt output pin = closed pin 2:3)
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	Select Button 1 depending on radio module (deRFarm7 or deRFmega128)

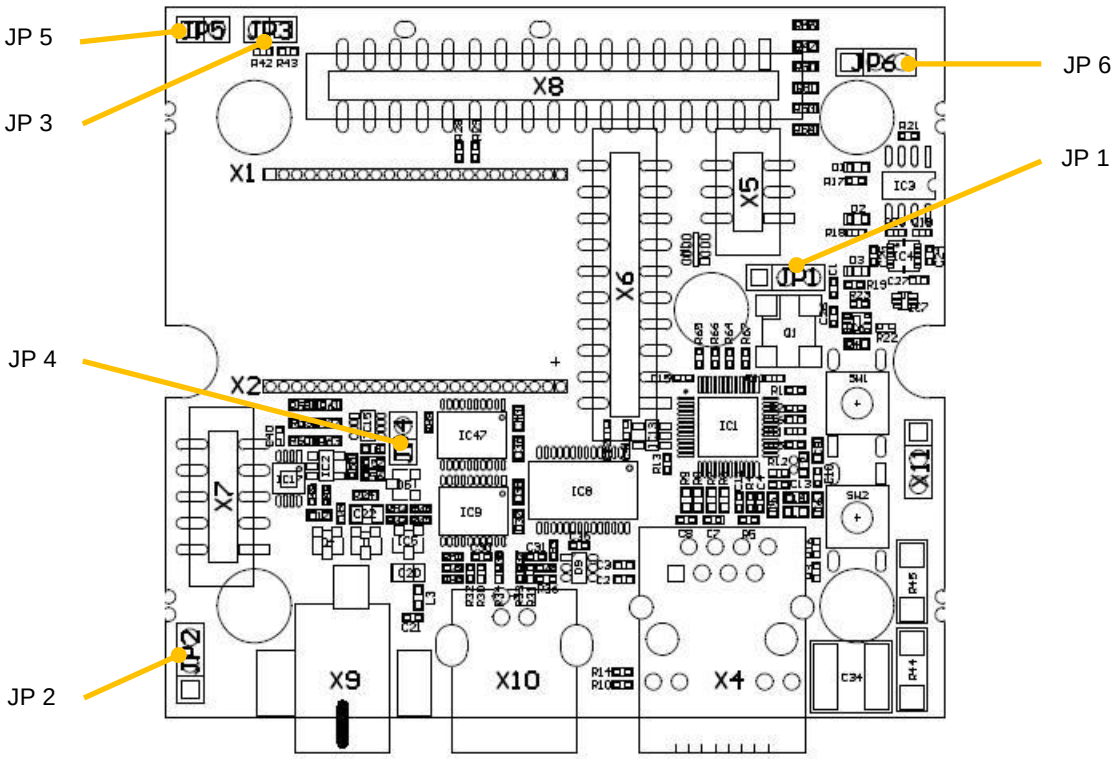


Figure 11: Jumper configuration

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.

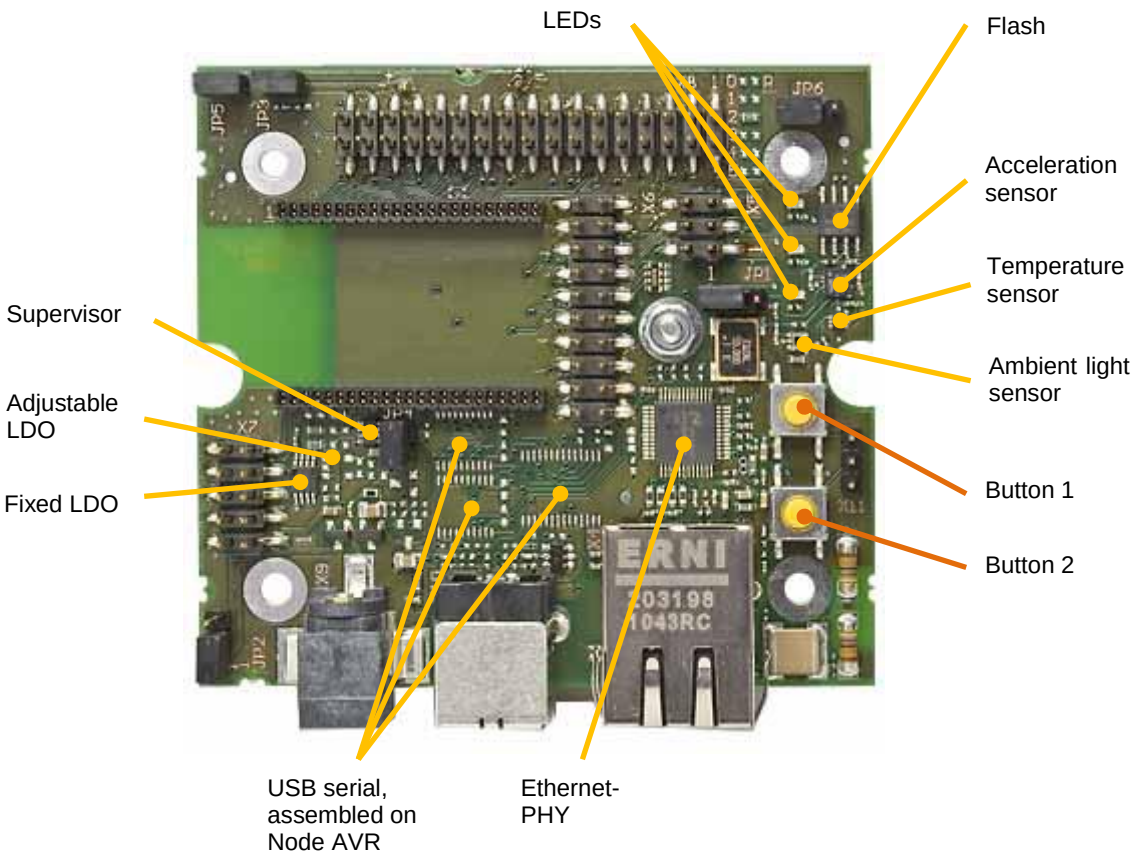


Figure 12: Board features

8.1. Onboard sensors

Both platforms are assembled with onboard sensors: temperature, ambient light and acceleration. All sensors are accessible over two-wire-interface of deRFnode and deRFgateway. The device addresses are noted in the following subsections. The data and clock lines are assembled 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.

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

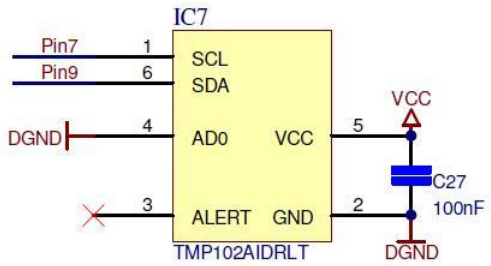


Figure 13: Temperature sensor TMP102AIDRLT

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.

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

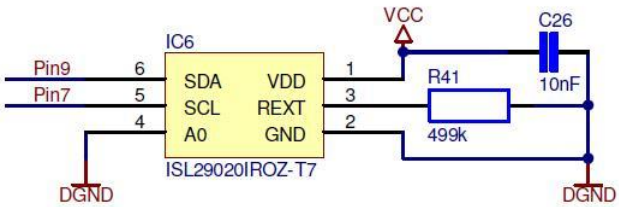


Figure 14: Ambient light sensor ISL29020IROZ-T7

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.

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

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

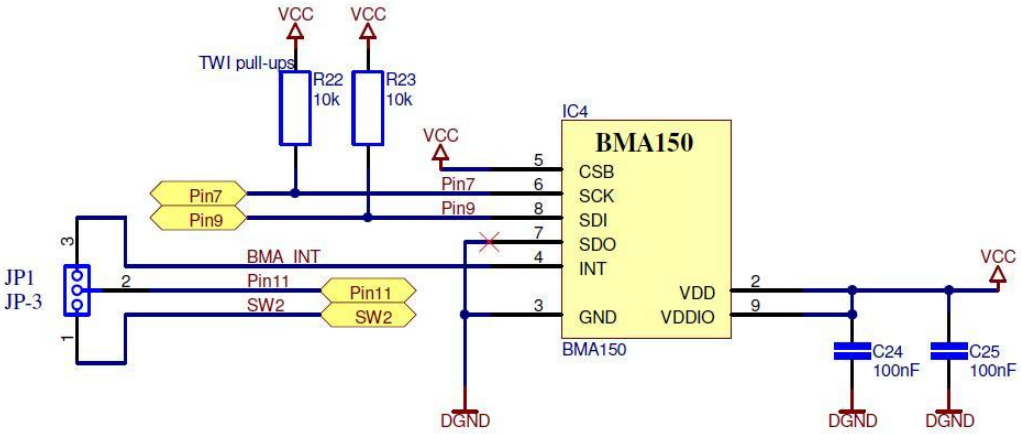


Figure 15: Acceleration sensor BMA150

8.2. LEDs and buttons

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

8.2.1. 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

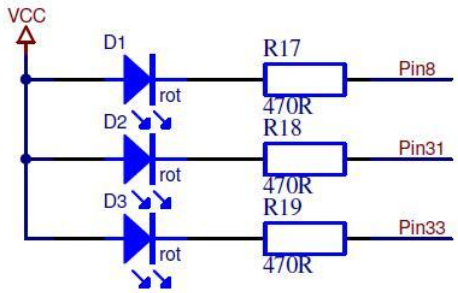


Figure 16: User LEDs

8.2.2. 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.

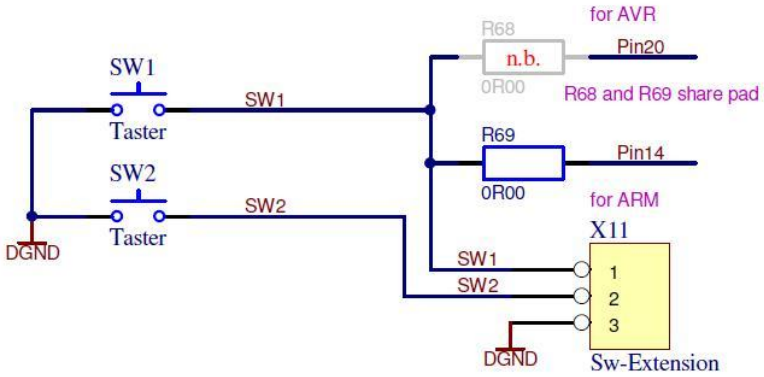


Figure 17: User buttons

8.3. USB interface

Regarding the USB interface, the platforms deRFnode and deRFgateway come in two different flavours. 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.

8.3.1. 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

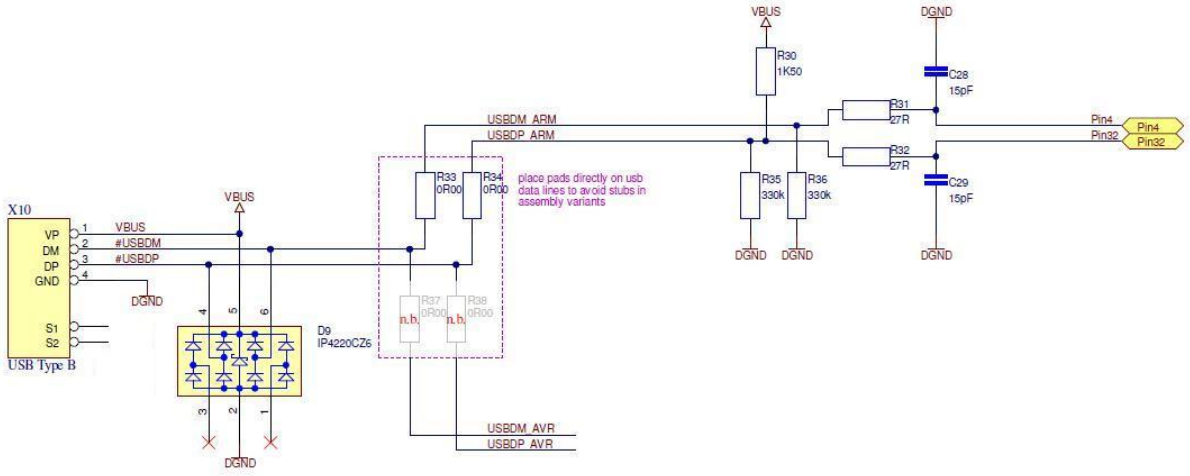


Figure 18: Native USB interface

8.3.2. 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 and the level shifter circuits TXB0108.

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

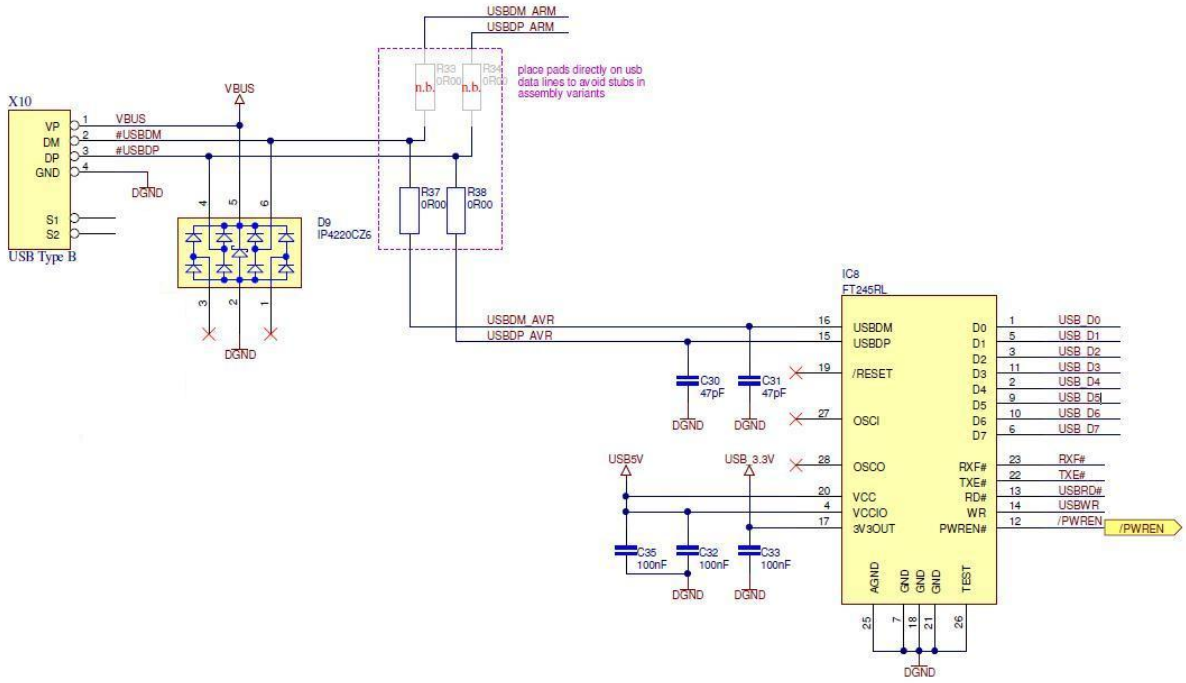


Figure 19: Serial USB interface part 1

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

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

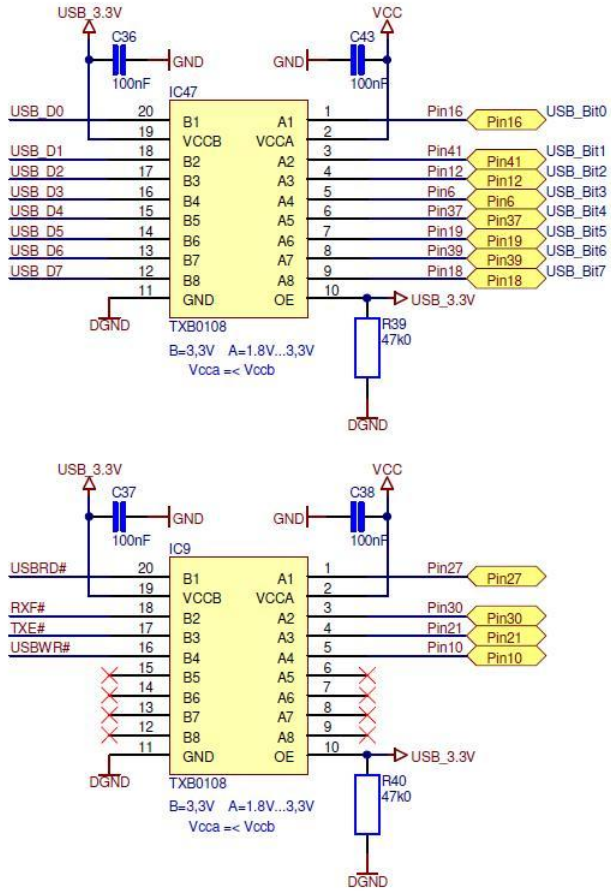


Figure 20: Serial USB interface part 2

8.4. 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 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
ETX1:	Pin 6
ERX0:	Pin 37
ERX1:	Pin 19
ETXEN:	Pin 41
ERXER:	Pin 39
ERXDV:	Pin 21
EMDC:	Pin 18
EMDIO:	Pin 30
MDINTR:	Pin 20

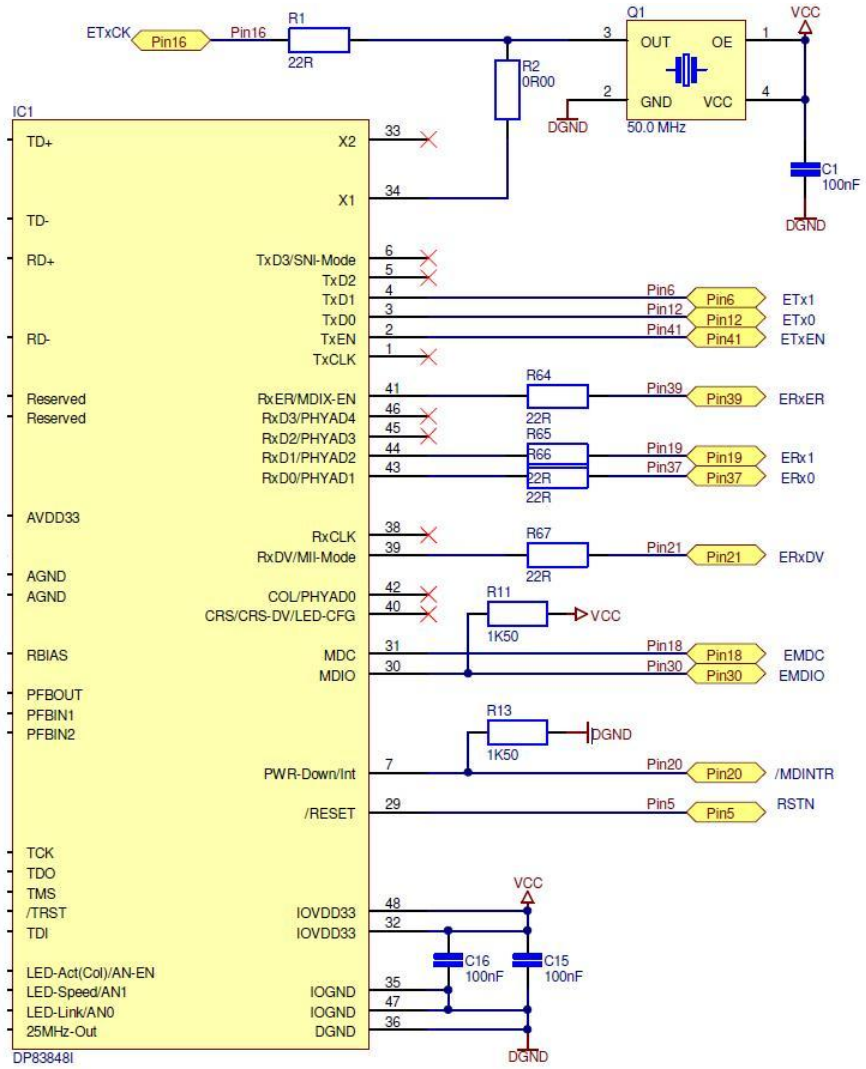


Figure 21: Ethernet-PHY DP83848I

8.5. 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].

Fixed LDO: TPS79433DGN
Adjustable LDO: TPS78001

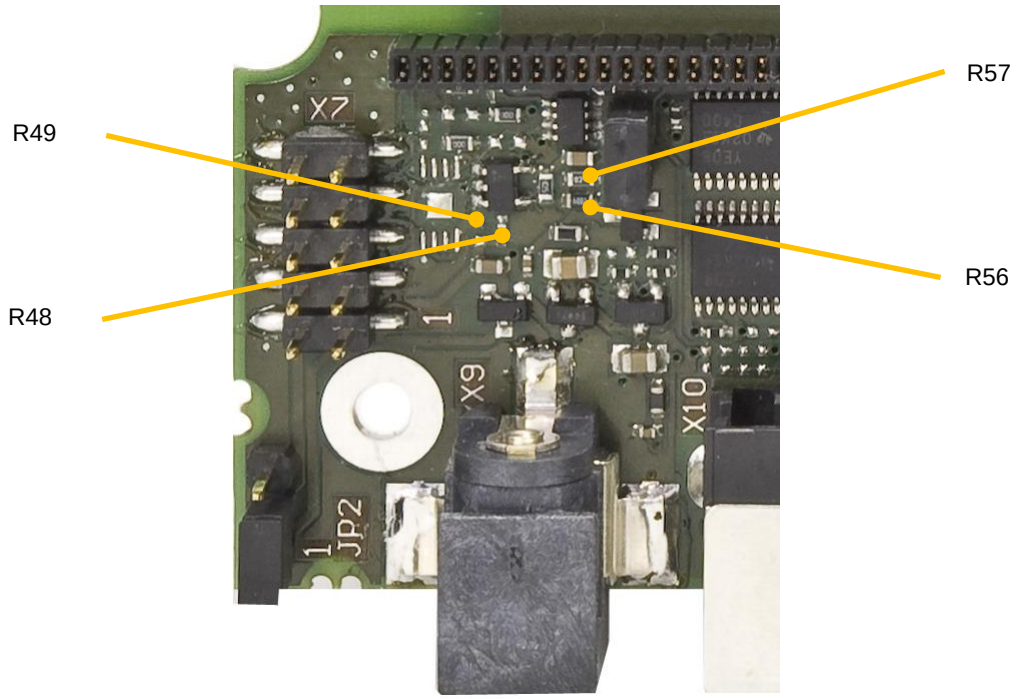


Figure 22: Resistors adjustable supply voltage

Table 16: 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 17: 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

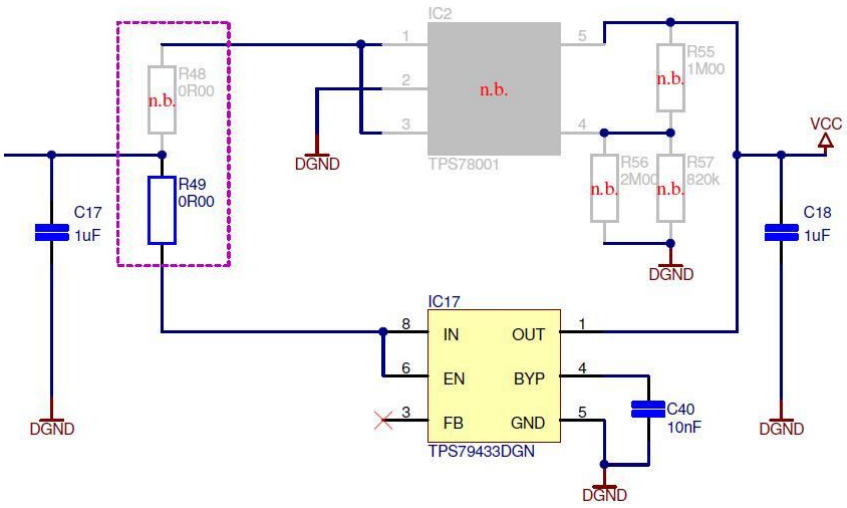


Figure 23: Fixed LDO TPS79433DGN

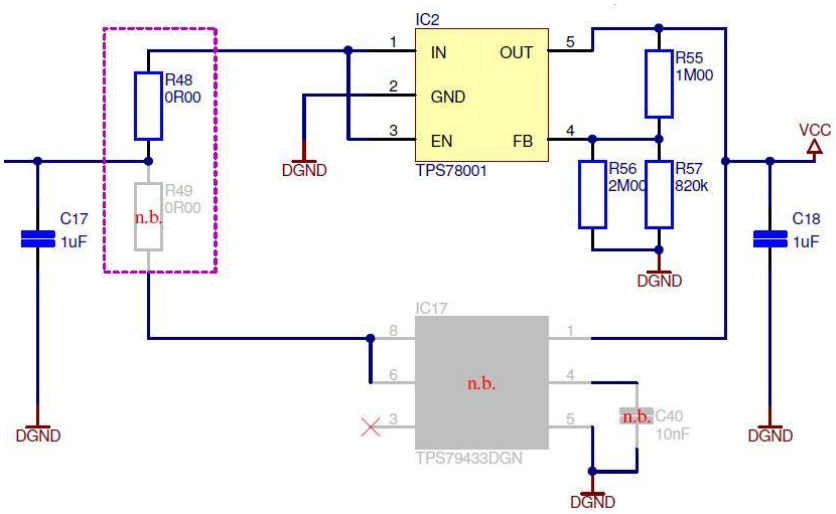


Figure 24: Adjustable LDO TPS78001

8.6. Supervisor

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

Reset: Pin 5

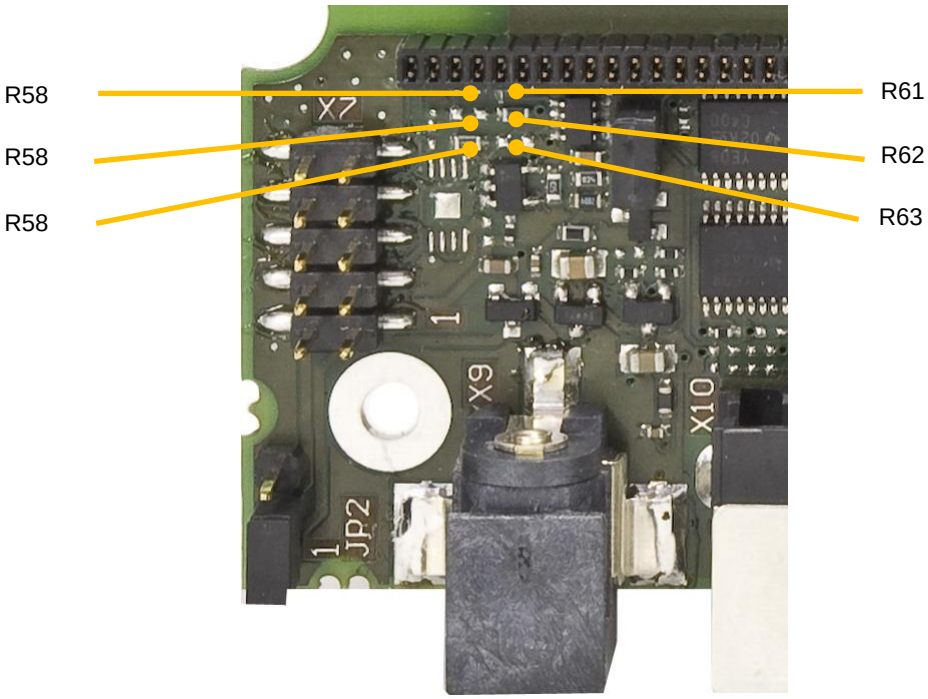


Figure 25: Resistors supervisor

Table 18: 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

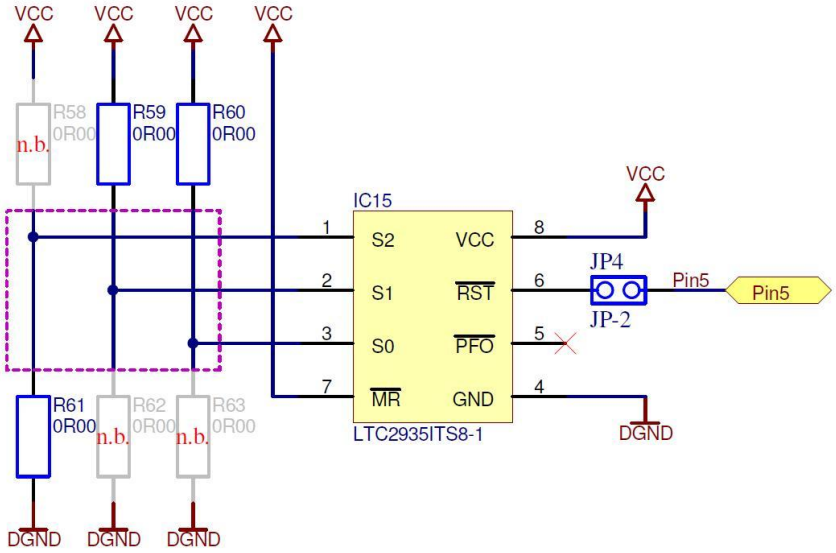


Figure 26: Supervisor LTC2935ITS8-1 with configuration resistors for deRFgateway

8.7. Current measurement

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

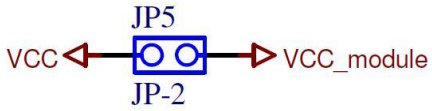


Figure 27: Current measurement for radio module

8.8. 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

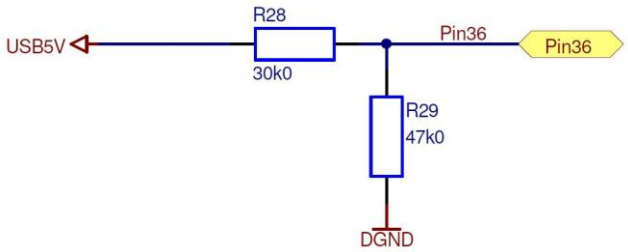


Figure 28: USB supply voltage monitoring

8.9. Battery supply voltage monitoring

It is useful to monitor the supply voltage of battery powered devices to detect the voltage level just before it is too low. 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

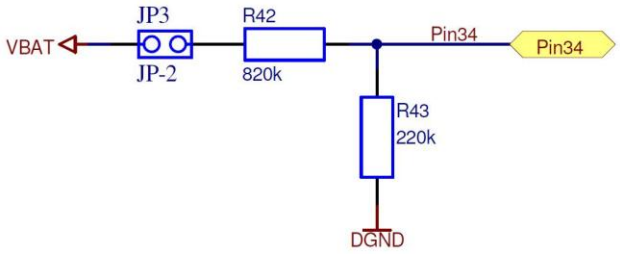


Figure 29: Battery supply voltage monitoring

9. Programming

9.1. Requirements (HW/SW)

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

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

Each of them is described in the following chapters. Dependent on the radio module used - either ARM-based (deRFarm7 series) or AVR-based (deRFmega128 series) - concrete recommendations will differ. Generally we suggest to use MS Windows™ as your developing platform, other operating systems are not supported yet.

The programming methods described below both base on using the JTAG interface. There will be a third programming option using the USB interface in near future.

9.1.1. 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 are required in addition. A good starting point for developing wireless software are the “Atmel IEEE 802.15.4 MAC Software Package” [1] or the “Atmel bitcloud“ [2]. Again 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 [3], arm-none-eabi-gcc comes with the yagarto-compiler package [4] and is extended by the yagarto tools [5]. 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.

9.1.2. JTAG programming and adapter selection

For AVR-based programming, we suggest to use the Atmel AVR ISP mkII Programming adapter. In case of an ARM-based MCU you may either use the Atmel AT91SAM-ICE JTAG Emulator [6], Segger J-Link [7], Amontec JTAGkey [8] or OpenOCD USB Adapter [9].

If you do not have a fully compatible combination of radio module and deRFnode/gateway-baseboard (refer to chapter 3), it may be required to upgrade the missing JTAG header and for ARM on AVR-boards also the 4x100K pull-up resistor network.

9.1.3. JTAG programming software

The required JTAG programming software differs depending on the chosen JTAG programming adapter. If you use winavr and a AVR ISP mkII, programming might be simply done by avrdude, a command-line tool which already comes with win-avr. If you prefer a GUI, take the free AVR-Studio from Atmel [10]. If you took SAM-ICE or the J-Link as programming adapter, it is recommended to use the SAM-Prog which is part of the SAM-BA programming software [11]. Amontecs and Olimex’ programming adapters may be accessed by Openocd [13].

9.2. AVR based radio modules over JTAG

This chapter exemplarily describes the programming process for a AVR based radio module put on a “deRFnode/gateway for AVR”. It is assumed that the customer uses a AVR ISP mkII as programming device, Atmel AVR Studio as programming software (already installed) and the desired firmware binary is accessible.

(1) Plug the programmer to the baseboard; ensure that the board is DC- or USB-powered.

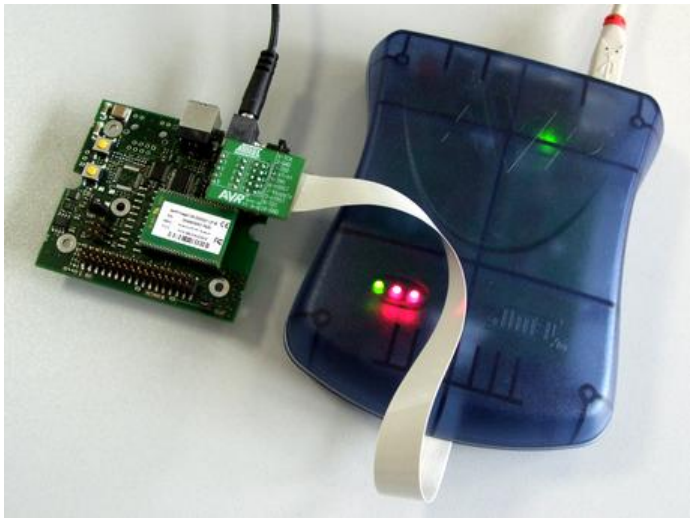


Figure 30: JTAGICE mkII plugged on deRFnode

(2) Open AVR Studio and connect to the programmer device (Tools|Program AVR|Connect).

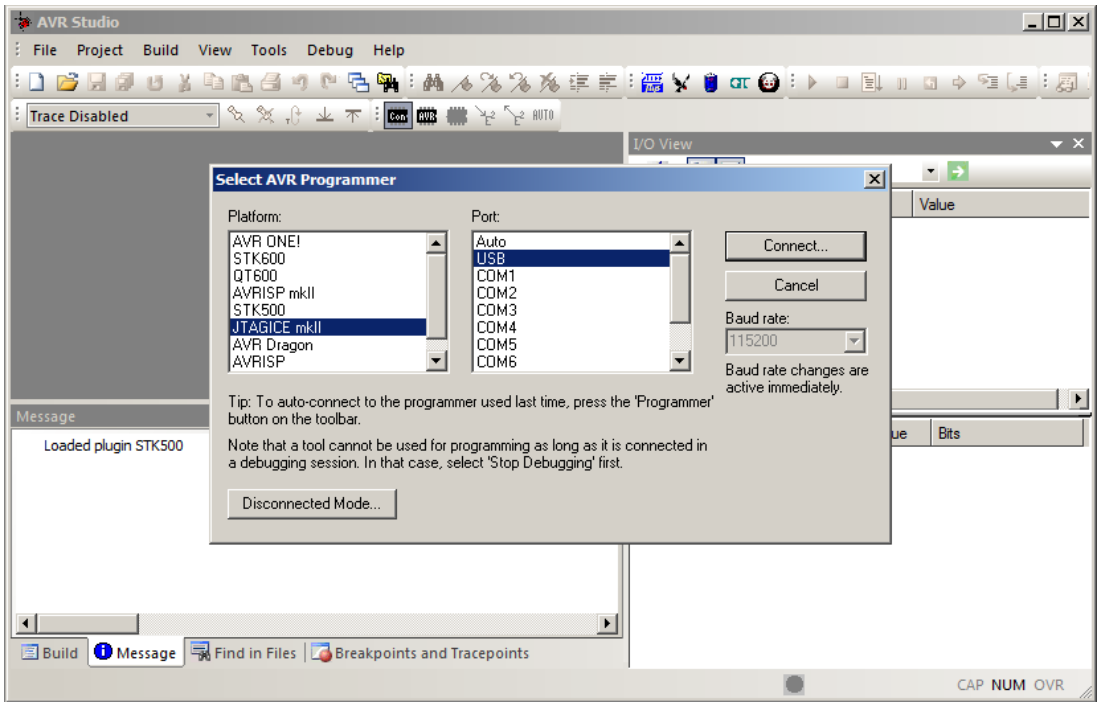


Figure 31: AVR Studio – select AVR programmer

(3) Try to identify the MCU, it must be an ATmega128RFA1.

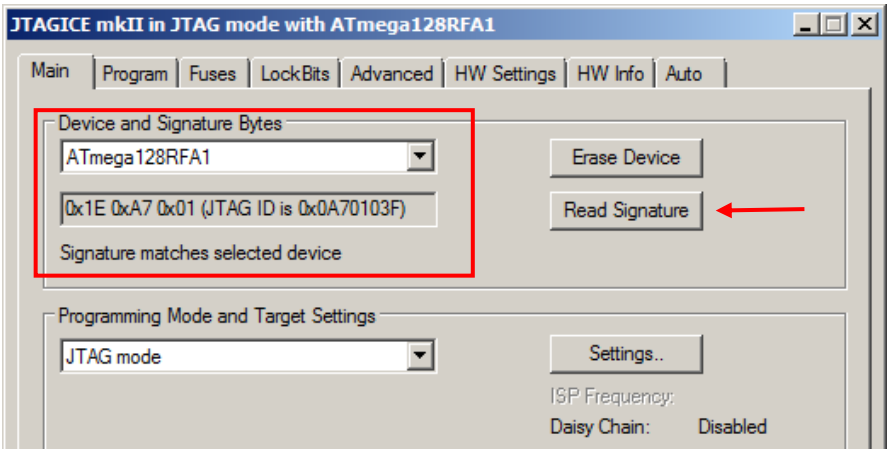


Figure 32: AVR Studio – select MCU

(4) Switch to 'program' tab, select the desired firmware binary, press 'Program' and wait until finished:

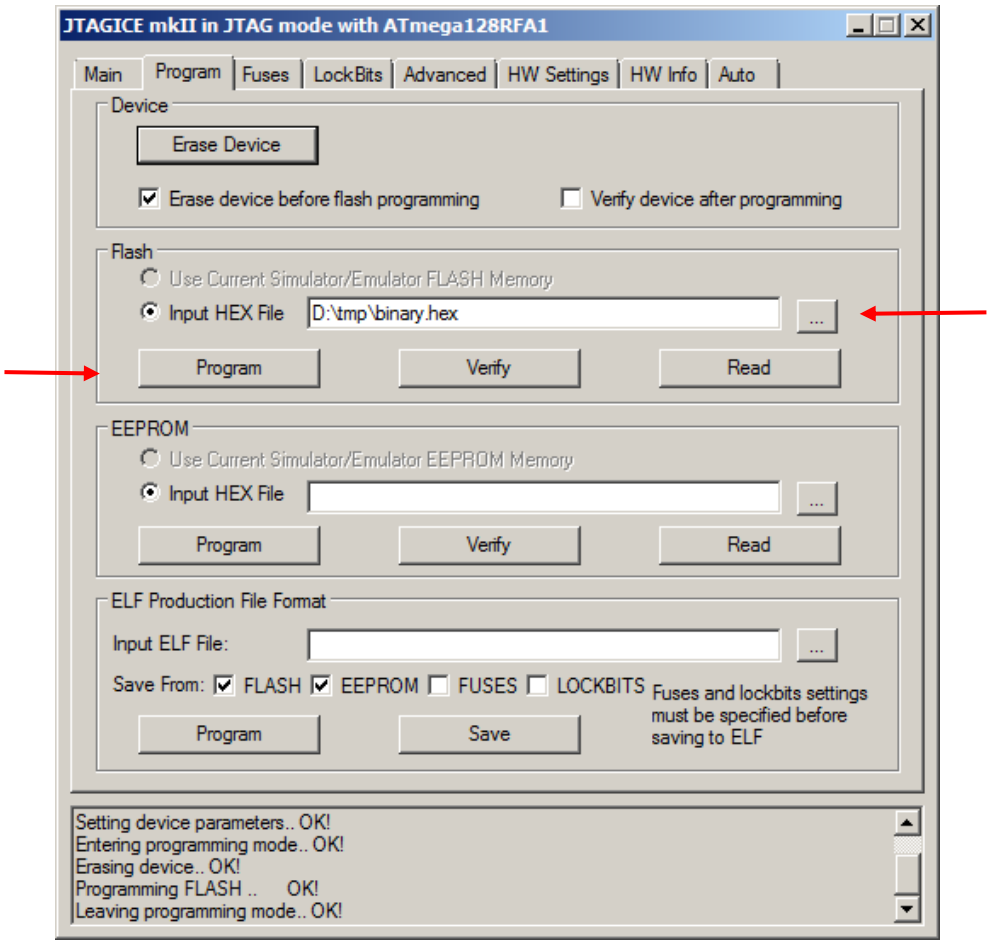


Figure 33: AVR Studio – program MCU

The programming procedure ends with a forced device reset after which the new firmware is executed.

9.3. ARM based radio modules over JTAG

In this chapter is described how to exchange the firmware of an ARM-based radio module using Atmel SAM-ICE JTAG Emulator and the SAM-Prog firmware updater which has already been installed. It is also assumed that the desired firmware binary was still compiled.

(1) Plug the programmer to the baseboard, ensure that the board is DC- or USB-powered.

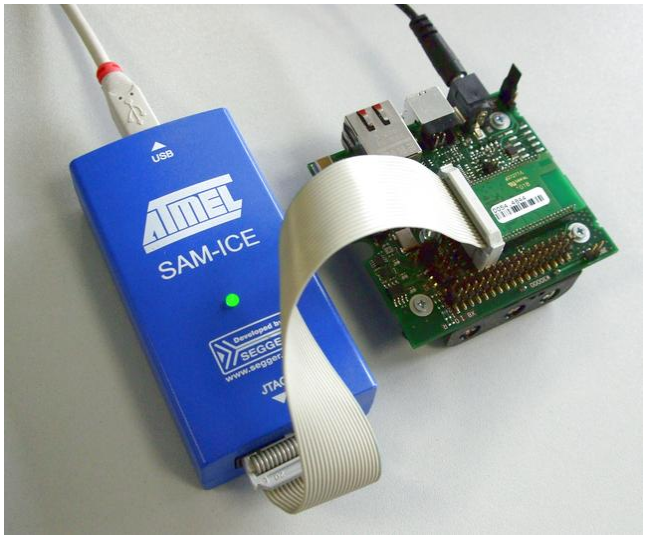


Figure 34: SAM-ICE plugged on deRFgateway

(2) Open SAM-Prog, choose the appropriate firmware file, then click 'Yes'. Be sure that the checkbox 'Auto-mode' is enabled. The firmware update process will be indicated by a flying sheet of paper.

(3) To complete the update procedure, restart the target device by temporary opening JP2 or removing power supply.

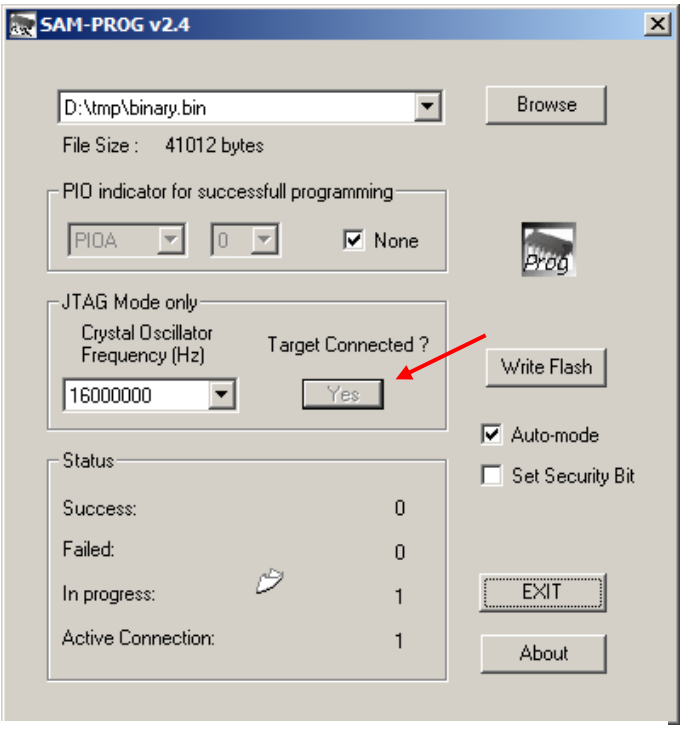


Figure 35: SAM-PROG

9.4. 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 [12] delivers besides many application examples the so called AT91Library, a large collection of higher-level-functions simplifying the development process. Since all following chapters base on it, 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 resting 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`.

9.4.1. 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.

9.4.2. Initialize and use I2C devices

All of the deRFnode/gateway boards include three environmental sensors which are accessible via the two-wire (aka I^2C)-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 enough for first tests while if excessive using the I2C interface, 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 I²C-library as well as out-of-the box working application examples for AVR and ARM which may be easily modified according to your needs.

9.4.3. 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 a 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 <<LINK>>. 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.

9.4.4. 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;
```

9.4.5. 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 such like:

```
/// 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.

9.4.6. 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 includes even this part.

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 KitCD will provide code examples in near future. 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.

9.4.7. Minimize device power consumption

Optimizing the energy consumption is especially important when 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 so in this case it makes no sense to think about power down modes. If finally Ethernet should be invoked, 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 chapter 4.

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 is 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 a 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:

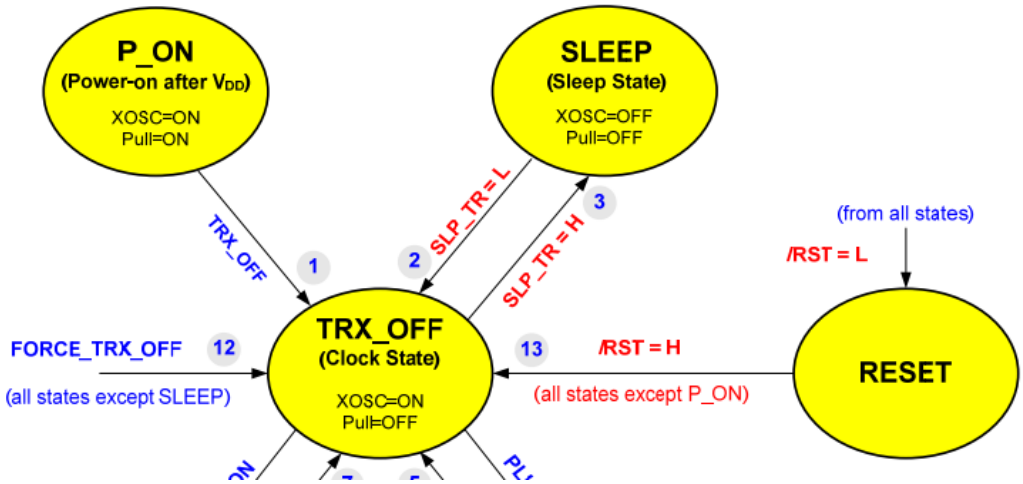


Figure 36: State control (source: ATMEL™ datasheet of AT86RF212 transceiver datasheet)

(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 left 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 achived 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.

10. Ordering information

The ordering code for deRFnode and deRFgateway are listed in Table 19.

Table 19: 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	
suitable case for deRFgateway and deRFnode	coming soon

11. Revision notes

Up to now for the deRFnode and deRFgateway platforms technical problems, malfunctions or any other critical issues are not known.

References

[1] Atmel IEEE 802.15.4 MAC Software Package revision 2.5.2
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4675

[2] Atmel BitCloud - ZigBee Pro
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4495

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

[4] yagarto GNU arm toolchain, version 20100813
http://www.yagarto.de/#download

[5] yagarto tools, version 20100703
http://www.yagarto.de/#download

[6] Atmel SAM-ICE JTAG Emulator
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3892

[7] Segger J-Link
http://www.segger.com/cms/jlink.html

[8] Amontec JTAGkey
www.amontec.com/jtagkey.shtml

[9] Olimex OpenOCD USB
http://www.olimex.com/dev/arm-usb-ocd.html

[10] AVR Studio 4.18
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[11] SAM-BA 2.10
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3883

[12] AT91SAM7X-512 Software package for IAR 5.2, Keil and GNU, revision 1.5
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4343
http://www.atmel.com/dyn/resources/prod_documents/at91sam7x-ek.zip

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

Tel. +49 351 - 31 85 00 | Fax +49 351 - 318 50 10
E-Mail wireless@dresden-elektronik.de
General manager: Dipl.-Ing. L. Pietschmann
Commercial Registry: HRB 749 Dresden Municipal Court
Tax number: 201/107/00726
Sales tax identification number: DE 140125678

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).
These 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.

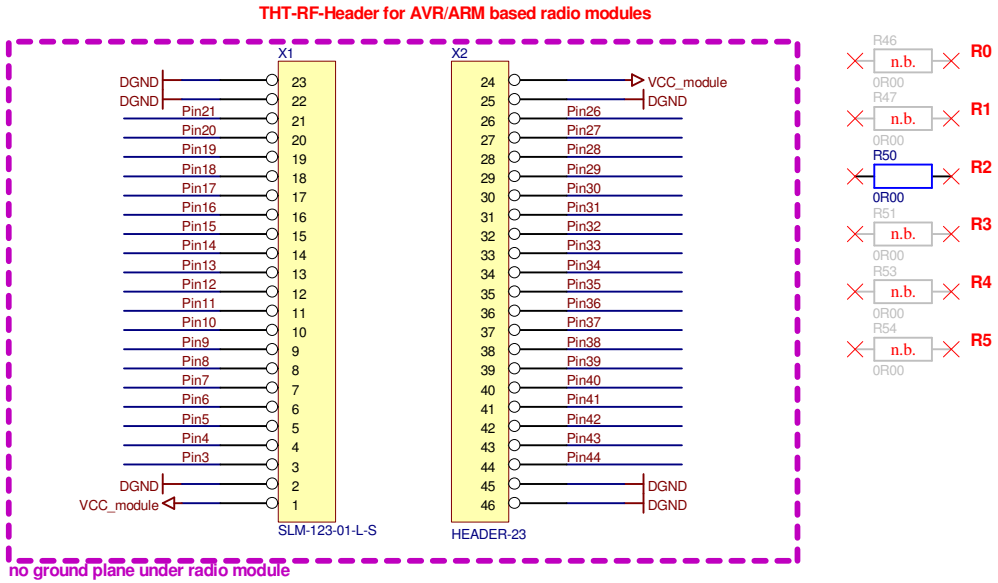
Disclaimer

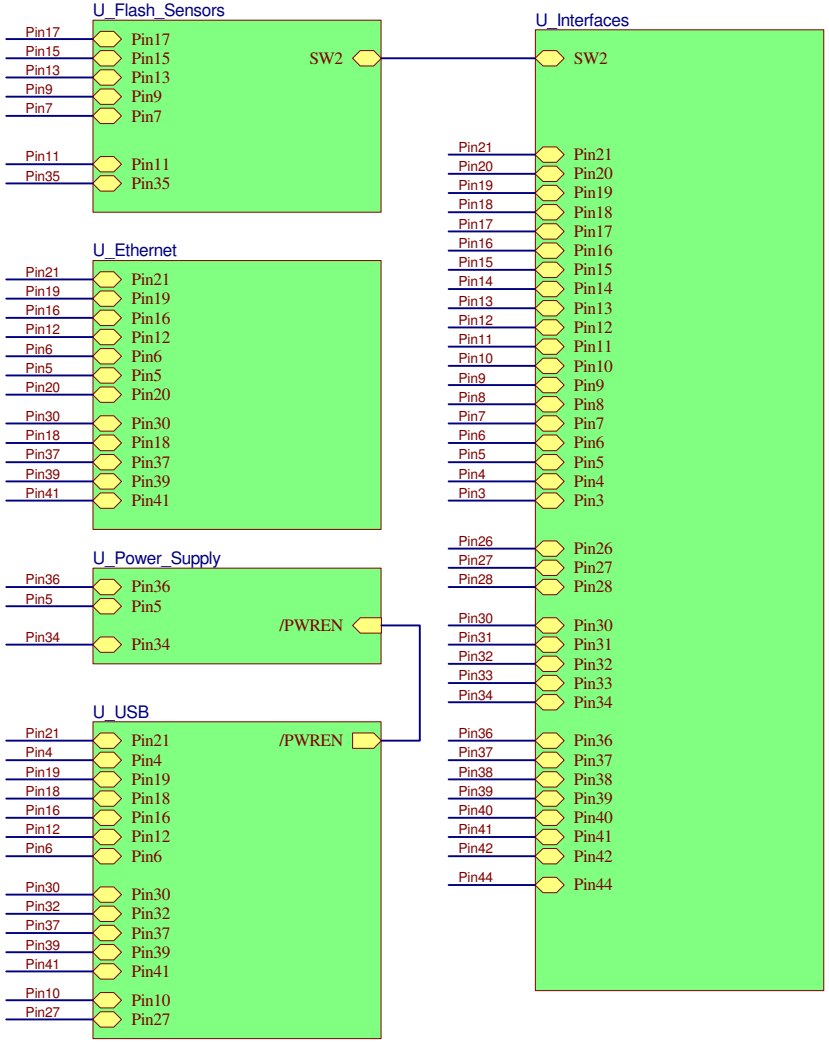
This note is provided as-is and is subject to change without notice. Except to the extent prohibited by law, dresden elektronik ingenieurtechnik gmbh makes no express or implied warranty of any kind with regard to this guide, and specifically disclaims the implied warranties and conditions of merchantability 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.
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use, without the written permission of dresden elektronik ingenieurtechnik gmbh.

Copyright © 2011 dresden elektronik ingenieurtechnik gmbh. All rights reserved

Annex A: Schematics

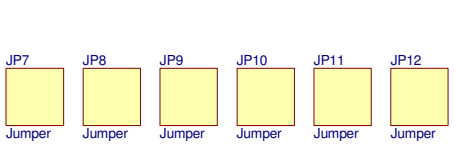
This section contains the schematics of deRFgateway-1TNP2, deRFnode-1TNP2 and deRFnode-2TNP2.

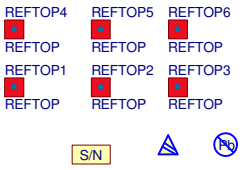




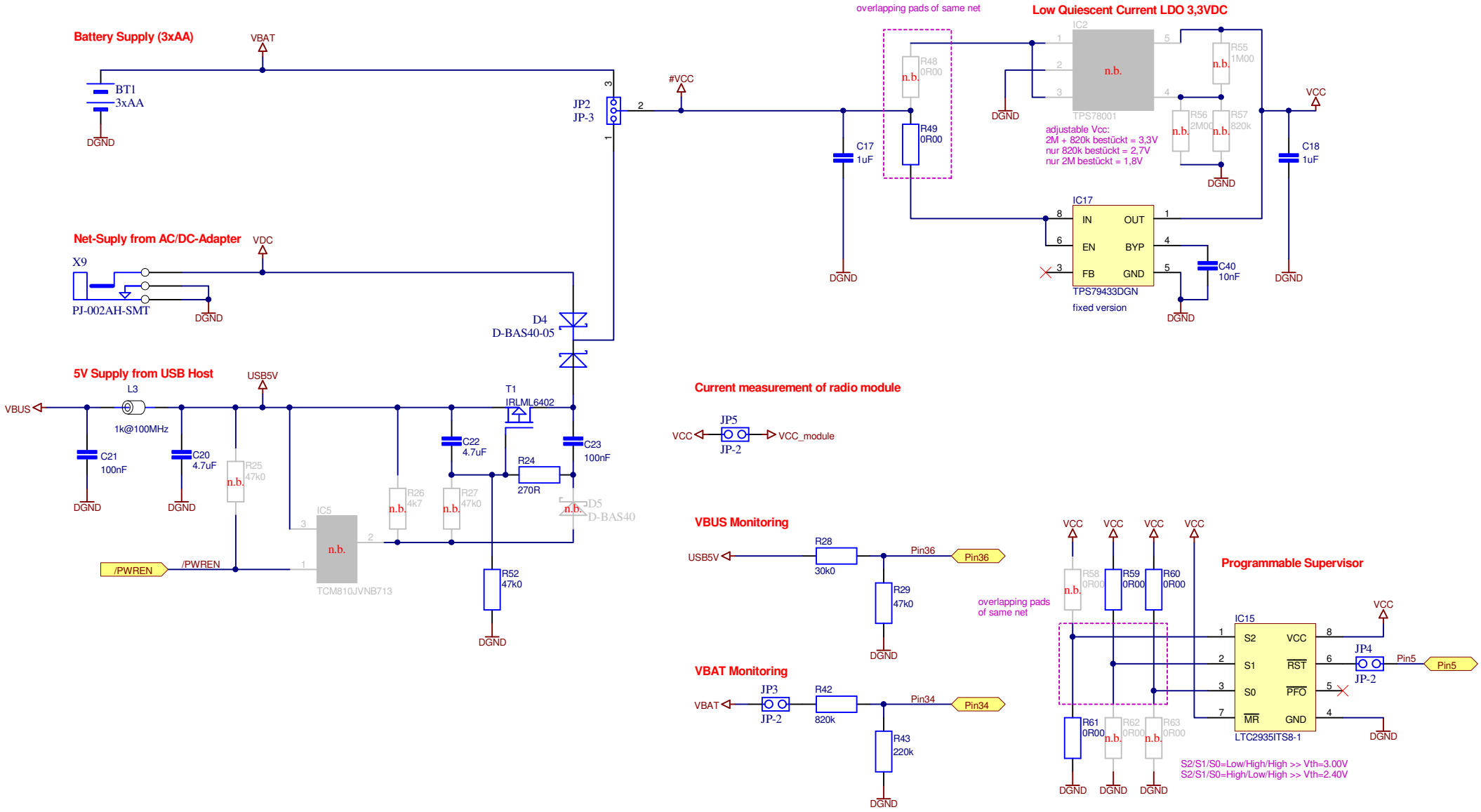
RF-Header Pin Description

Port	deRFmega128 ATmega128RFA1	deRFarm7-x5xx1 AT91SAM7X512-CU	Port	deRFmega128 ATmega128RFA1	deRFarm7-x5xx1 AT91SAM7X512-CU
Pin23	GND	GND	Pin24	VCC	VCC
Pin22	GND	GND	Pin25	GND	GND
Pin21	PB5/OC1A/PCINT5	PB15/ERXDV/ECRSDV	Pin26	PE0/RXD0/PCINT8	PA27/DRXD/PCK3
Pin20	PB7/OC0A/OC1C/PCINT7	PB18/EF100/ADTRG	Pin27	PD2/RXD1/INT2	PA0/RXD0
Pin19	PB4/OC2/PCINT4	PB6/ERX1	Pin28	PE1/TXD0	PA28/DTXD
Pin18	PB6/OC1B/PCINT6	PB8/EMDC	Pin29	PD6/T1	PA4/CTS0/SPI1_NPCS3
Pin17	PB3/MISO/PCINT3/PDO	PA16/SPI0_MISO	Pin30	PE2/XCK0/AIN0	PB9/EMDIO
Pin16	PB0/SSN/PCINT0	PB0/ETXCK/EREFCK	Pin31	PE3/OC3A/AIN1	PB21/PWM2/PCK1
Pin15	PB2/MOSI/PCINT2/PDI	PA17/SPI0_MOSI	Pin32	PD4/ICP1	USBDP
Pin14	CLKI	PA3/RTS0/SPI1_NPCS2	Pin33	PE4/OC3B/INT4	PB19/PWM0/TCLK1
Pin13	PB1/SCK/PCINT1	PA18/SPI0_SPCK	Pin34	PF0/ADC0	PB27/TIOA2/PWM0/AD0
Pin12	PD5/XCK1	PB2/ETX0	Pin35	PE5/OC3C/INT5	PA14/SPI0_NPCS2/IRQ1
Pin11	PD7/T0	PB25/TIOA1/DTR1	Pin36	PF1/ADC1	PB28/TIOB2/PWM1/AD1
Pin10	PD3/TXD1/INT3	PA1/ TXD0	Pin37	PE6/T3/INT6	PB5/ERX0
Pin9	PD1/SDA/INT1	PA10/TWD	Pin38	PF4/ADC4/TCK	TCK
Pin8	PG5/OC0B	PB26/TIOB1/RI1	Pin39	PE7/ICP3/CLKO/INT7	PB7/ERXER
Pin7	PD0/SCL/INT0	PA11/TWCK	Pin40	PF5/ADC5/TMS	TMS
Pin6	PG2	PB3/ETX1	Pin41	PF2/ADC2	PB1/ETXEN
Pin5	RSTN	RSTN	Pin42	PF6/ADC6/TDO	TDO
Pin4	PG1/DIG1	USBDM	Pin43	RSTON	JTAGSEL
Pin3	AREF	ADVREF	Pin44	PF7/ADC7/TDI	TDI
Pin2	GND	GND	Pin45	GND	GND
Pin1	VCC	VCC	Pin46	GND	GND

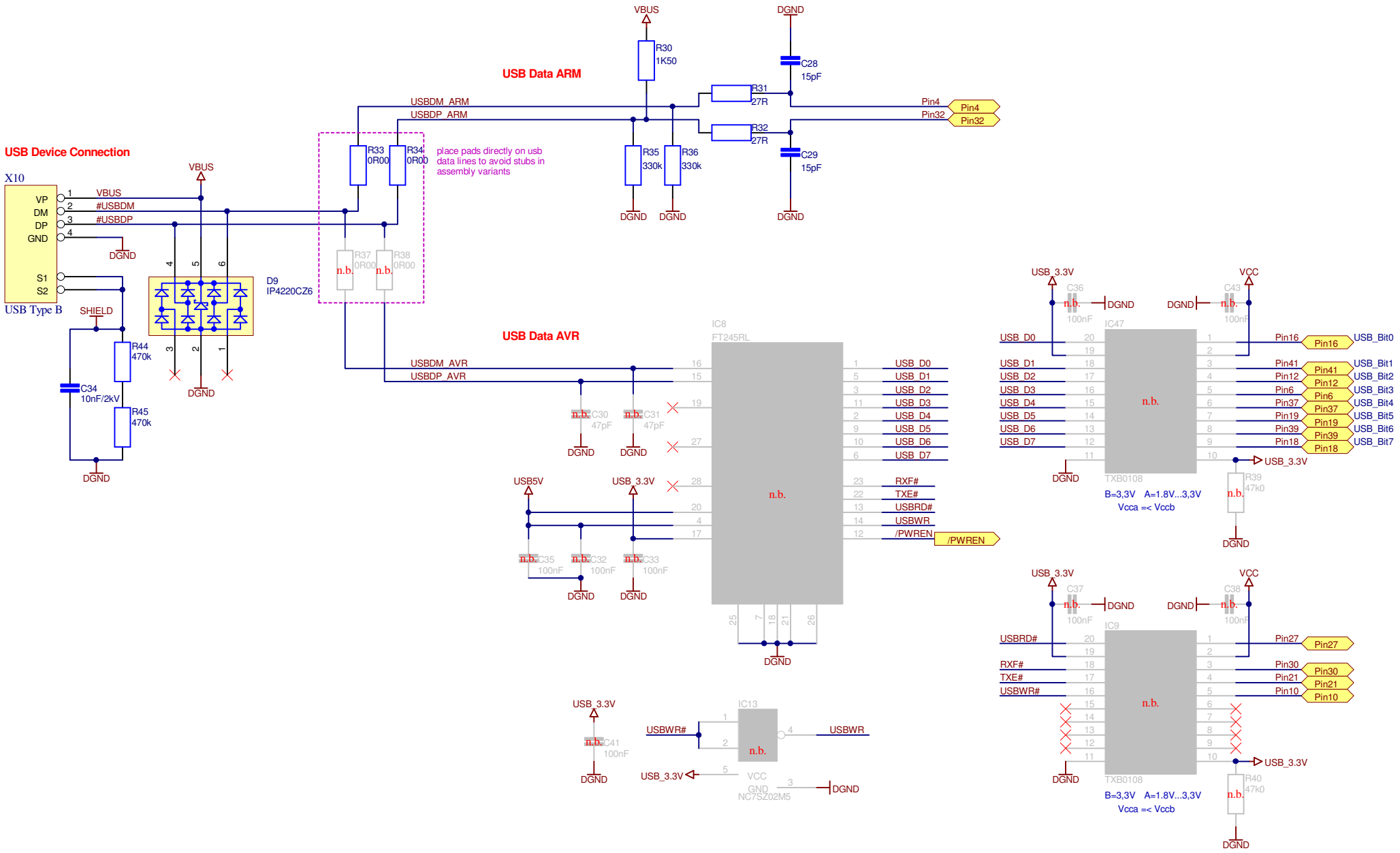




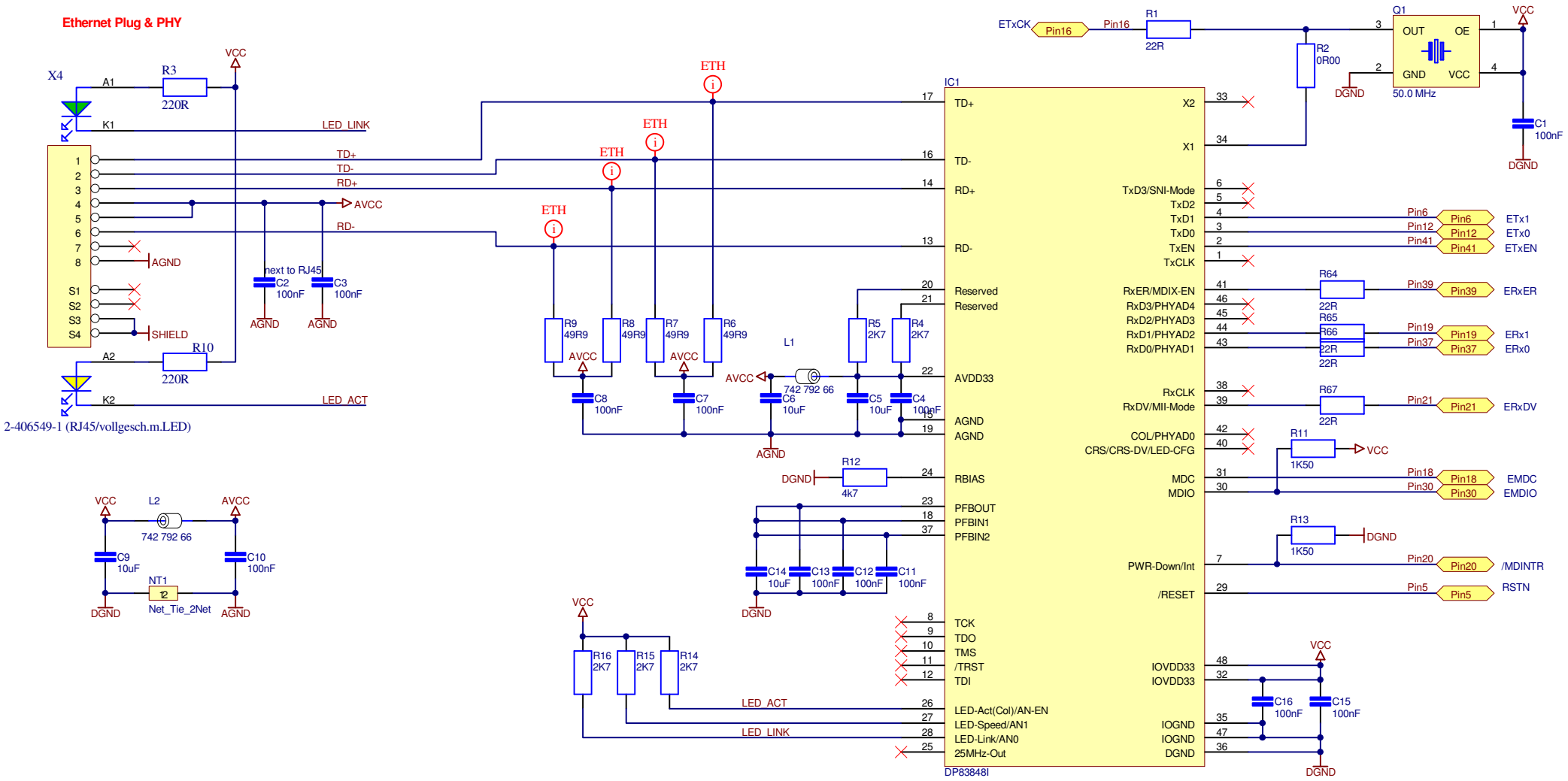
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title deRFgateway-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.103.02		Document File Name Module_Connectors.SchDoc
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:24	DrawnBy: NOS/APA	Sheet 1 of 6 | Rev: 2.0



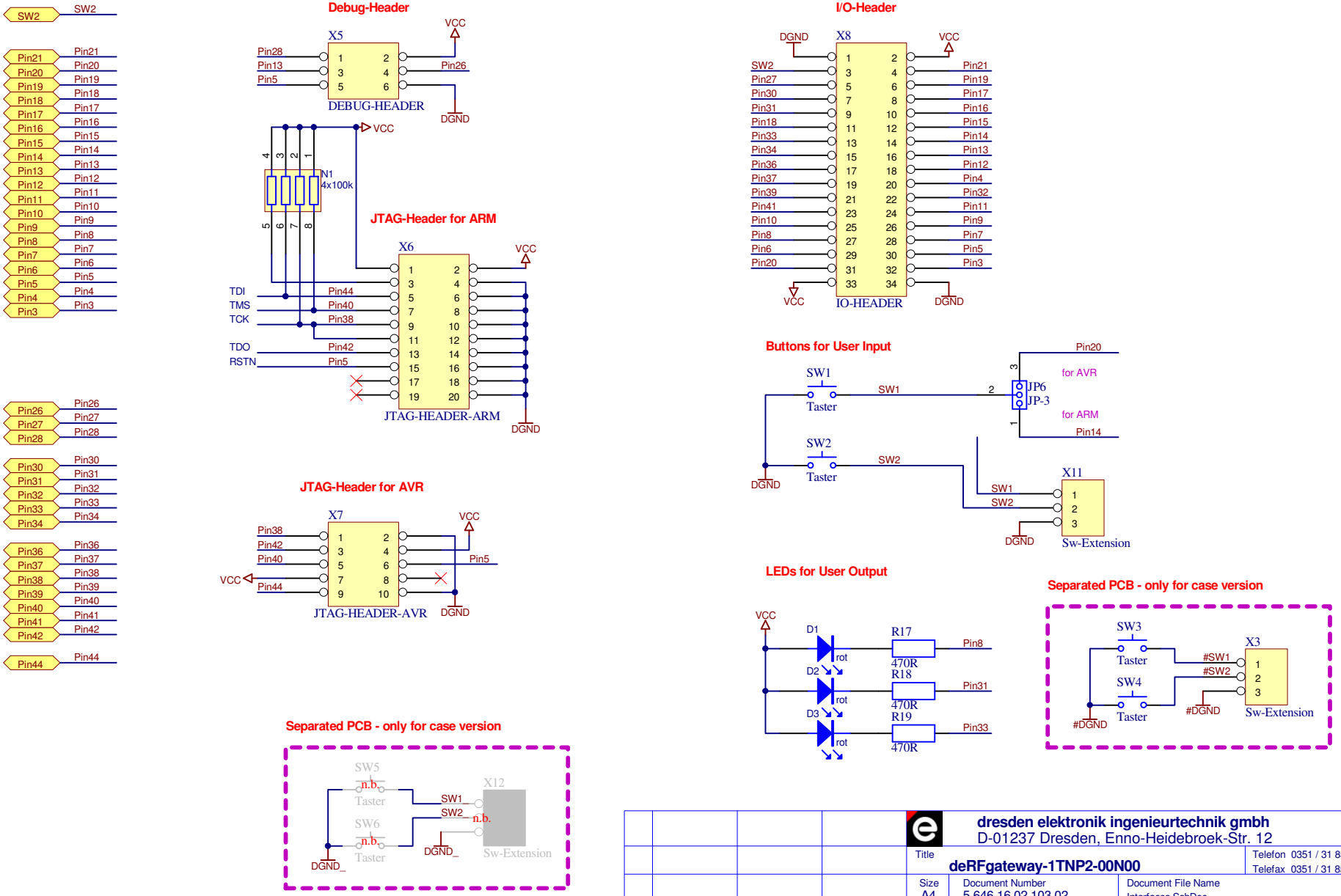
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFgateway-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.103.02		Document File Name Power_Supply.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:25	DrawnBy: NOS/APA	Sheet 2 of 6	Rev: 2.0



				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFgateway-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.103.02	Document File Name USB.SchDoc		
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:25	DrawnBy: NOS/APA	Sheet 3 of 6	Rev: 2.0

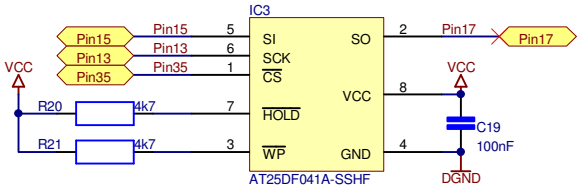


				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFgateway-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.103.02	Document File Name Ethernet.SchDoc		
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:25	DrawnBy: NOS/APA	Sheet 4 of 6	Rev: 2.0



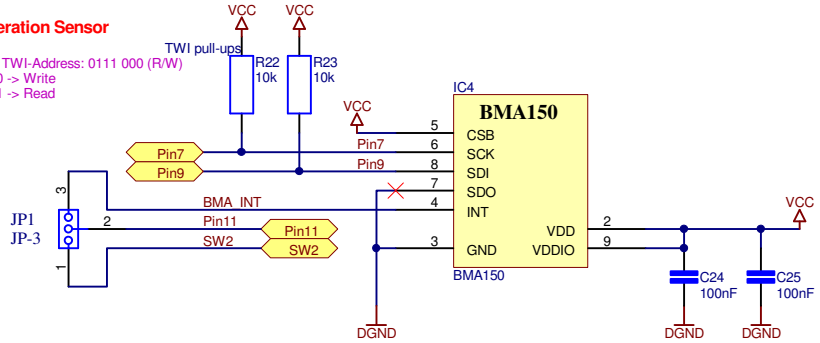
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title deRFgateway-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.103.02	Document File Name Interfaces.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:25	DrawnBy: NOS/APA	Sheet 5 of 6 Rev: 2.0

4M Bit Flash Memory



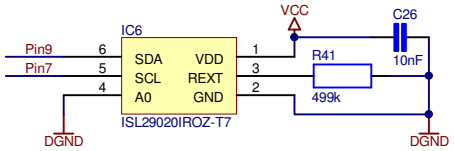
Acceleration Sensor

Internal TWI-Address: 0111 000 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



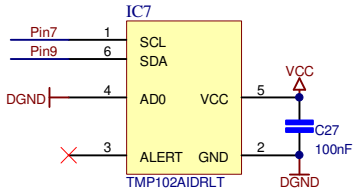
Ambient Light Sensor

Internal TWI-Address: 1000 100 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read

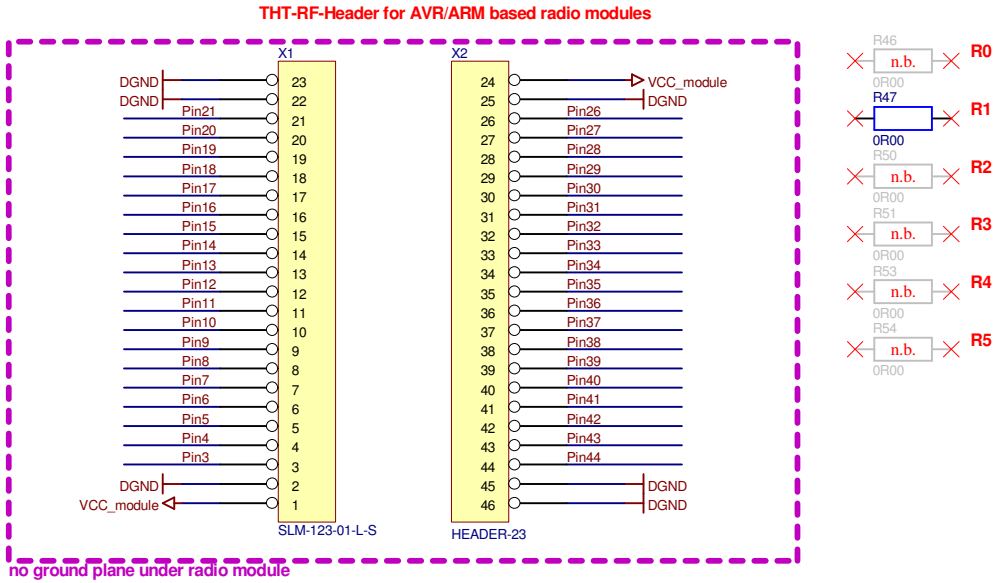


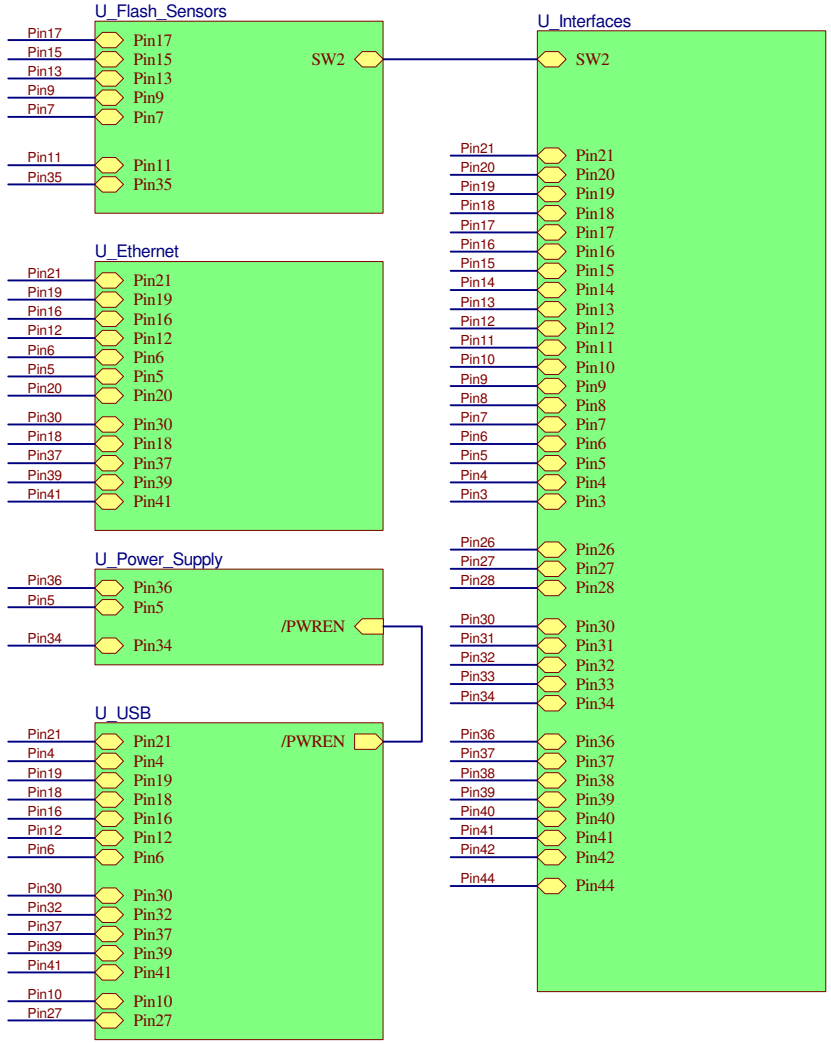
Temperature Sensor

Internal TWI-Address: 1001 000 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



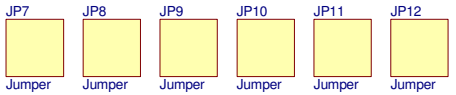
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFgateway-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.103.02		Document File Name Flash_Sensors.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011		Time: 10:13:26	DrawnBy: NOS/APA	Sheet 6 of 6 Rev: 2.0

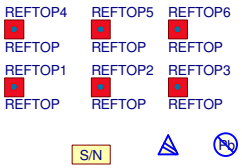




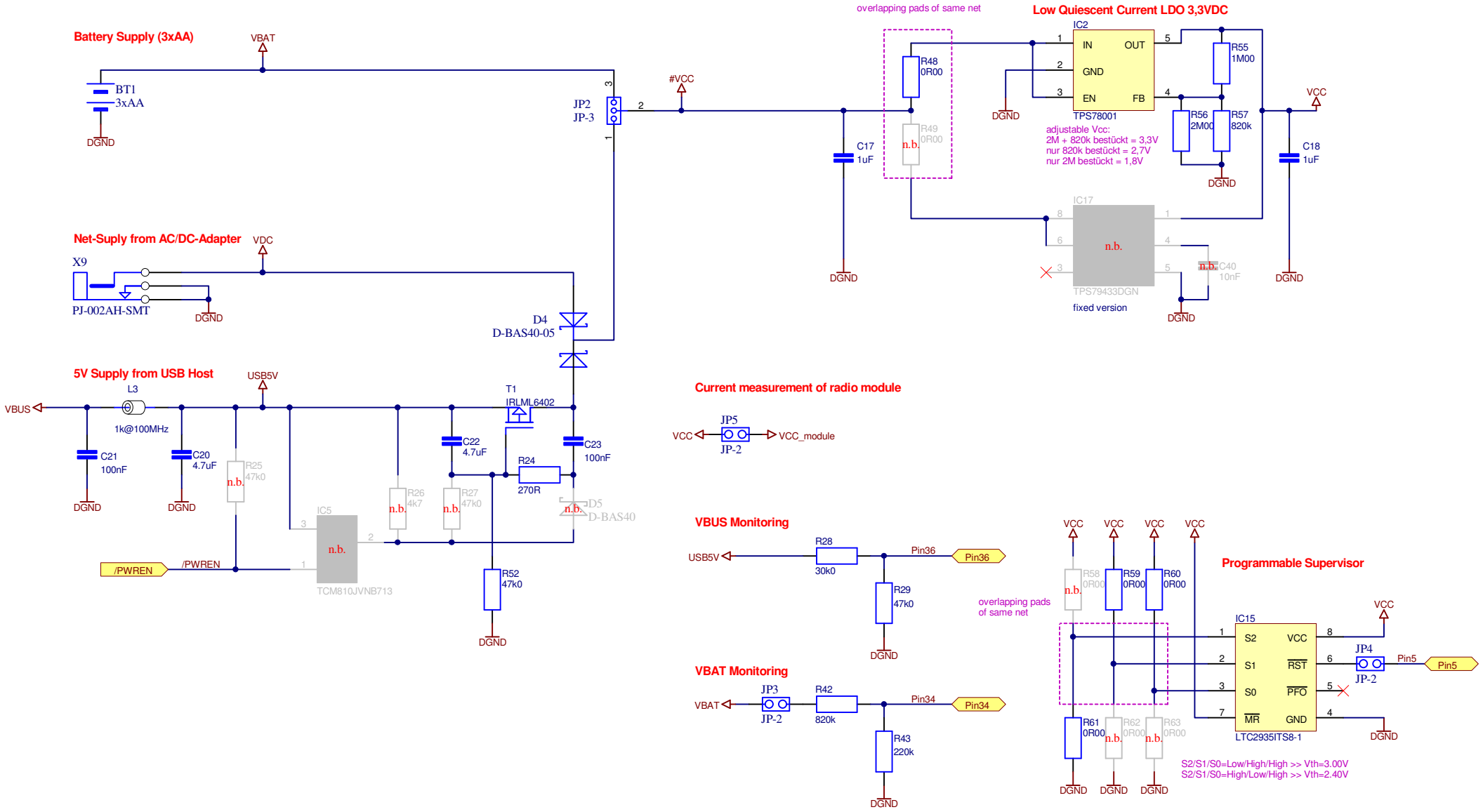
RF-Header Pin Description

Port	deRFmega128 ATmega128RFA1	deRFarm7-x5xx1 AT91SAM7X512-CU	Port	deRFmega128 ATmega128RFA1	deRFarm7-x5xx1 AT91SAM7X512-CU
Pin23	GND	GND	Pin24	VCC	VCC
Pin22	GND	GND	Pin25	GND	GND
Pin21	PB5/OC1A/PCINT5	PB15/ERXDV/ECRSDV	Pin26	PE0/RXD0/PCINT8	PA27/DRXD/PCK3
Pin20	PB7/OC0A/OC1C/PCINT7	PB18/EF100/ADTRG	Pin27	PD2/RXD1/INT2	PA0/RXD0
Pin19	PB4/OC2/PCINT4	PB6/ERX1	Pin28	PE1/TXD0	PA28/DTXD
Pin18	PB6/OC1B/PCINT6	PB8/EMDC	Pin29	PD6/T1	PA4/CTS0/SPI1_NPCS3
Pin17	PB3/MISO/PCINT3/PDO	PA16/SPI0_MISO	Pin30	PE2/XCK0/AIN0	PB9/EMDIO
Pin16	PB0/SSN/PCINT0	PB0/ETXCK/EREFCK	Pin31	PE3/OC3A/AIN1	PB21/PWM2/PCK1
Pin15	PB2/MOSI/PCINT2/PDI	PA17/SPI0_MOSI	Pin32	PD4/ICP1	USBDP
Pin14	CLKI	PA3/RTS0/SPI1_NPCS2	Pin33	PE4/OC3B/INT4	PB19/PWM0/TCLK1
Pin13	PB1/SCK/PCINT1	PA18/SPI0_SPCK	Pin34	PF0/ADC0	PB27/TIOA2/PWM0/AD0
Pin12	PD5/XCK1	PB2/ETX0	Pin35	PE5/OC3C/INT5	PA14/SPI0_NPCS2/IRQ1
Pin11	PD7/T0	PB25/TIOA1/DTR1	Pin36	PF1/ADC1	PB28/TIOB2/PWM1/AD1
Pin10	PD3/TXD1/INT3	PA1/ TXD0	Pin37	PE6/T3/INT6	PB5/ERX0
Pin9	PD1/SDA/INT1	PA10/TWD	Pin38	PF4/ADC4/TCK	TCK
Pin8	PG5/OC0B	PB26/TIOB1/RI1	Pin39	PE7/ICP3/CLKO/INT7	PB7/ERXER
Pin7	PD0/SCL/INT0	PA11/TWCK	Pin40	PF5/ADC5/TMS	TMS
Pin6	PG2	PB3/ETX1	Pin41	PF2/ADC2	PB1/ETXEN
Pin5	RSTN	RSTN	Pin42	PF6/ADC6/TDO	TDO
Pin4	PG1/DIG1	USBDM	Pin43	RSTON	JTAGSEL
Pin3	AREF	ADVREF	Pin44	PF7/ADC7/TDI	TDI
Pin2	GND	GND	Pin45	GND	GND
Pin1	VCC	VCC	Pin46	GND	GND

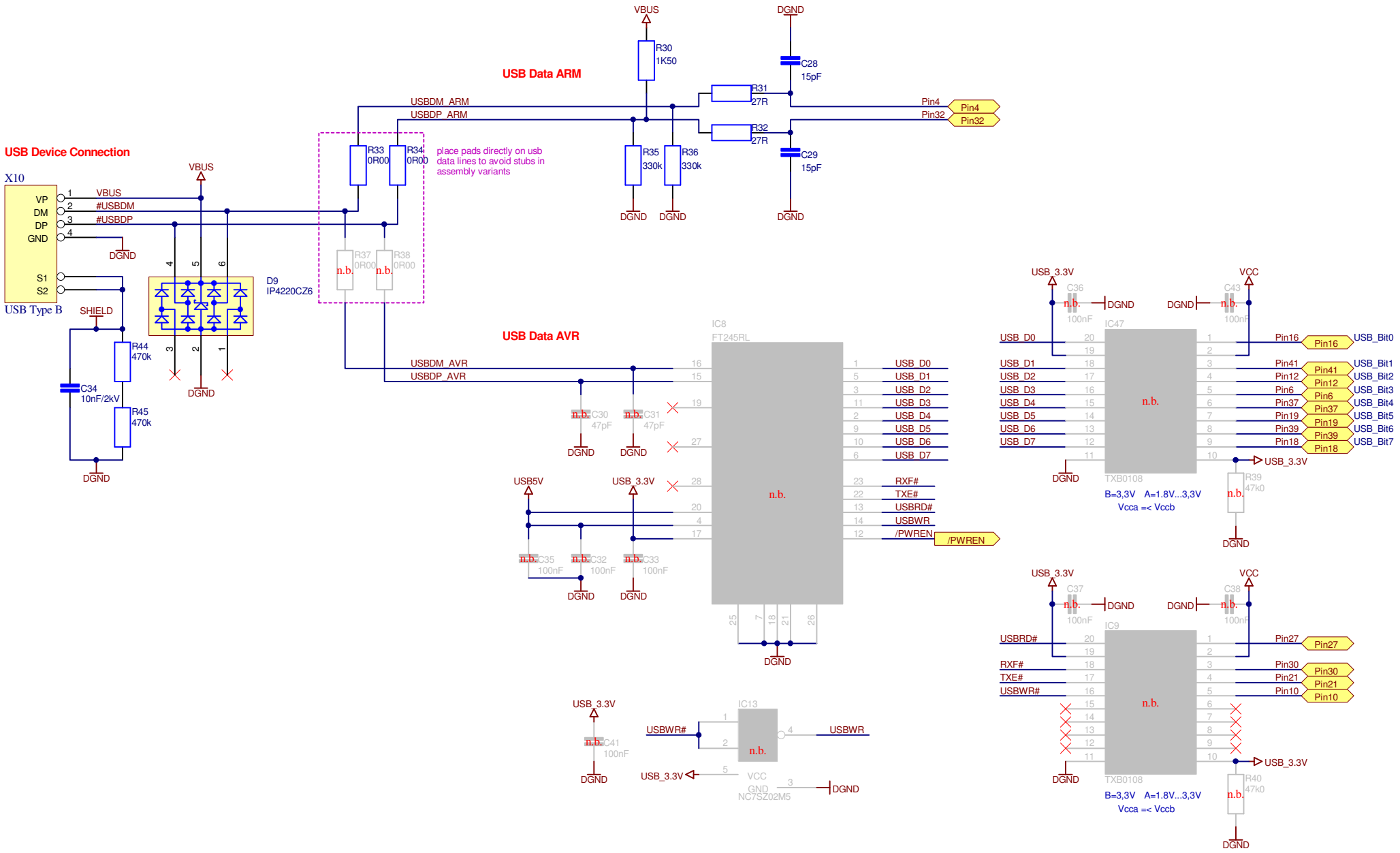




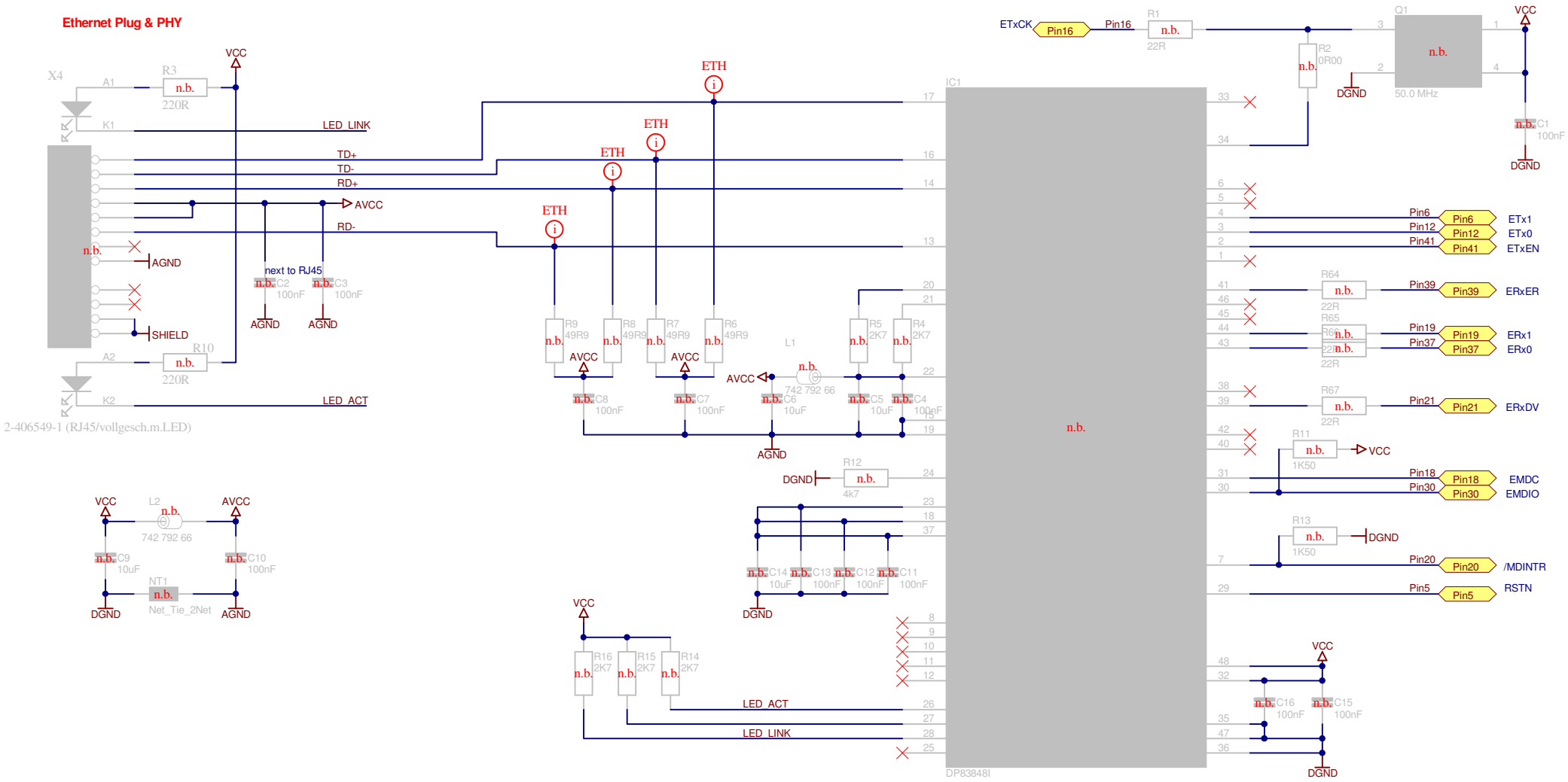
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12		
				Title: deRFnode-1TNP2-00N00		Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size: A4	Document Number: 5 646 16 02.101.02	Document File Name: Module_Connectors.SchDoc
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:44	DrawnBy: NOS/APA / Sheet 1 of 6 / Rev: 2.0



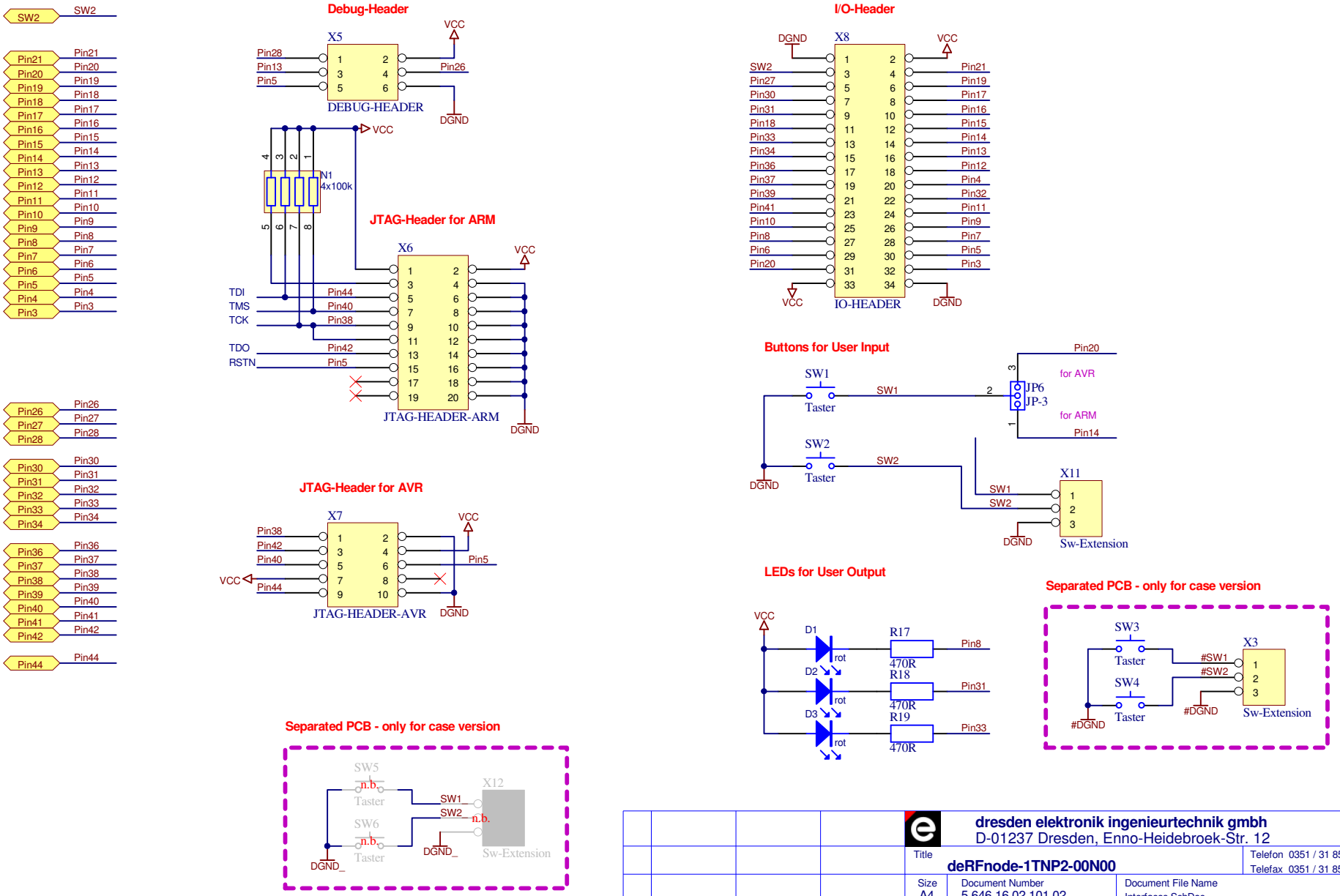
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title deRFnode-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.101.02	Document File Name Power_Supply.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:44	DrawnBy: NOS/APA	Sheet 2 of 6 Rev: 2.0



				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFnode-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.101.02		Document File Name USB.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:44	DrawnBy: NOS/APA	Sheet 3 of 6	Rev: 2.0

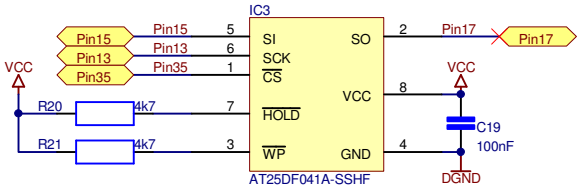


				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFnode-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.101.02		Document File Name Ethernet.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011		Time: 10:13:44	DrawnBy: NOS/APA	Sheet 4 of 6 Rev: 2.0



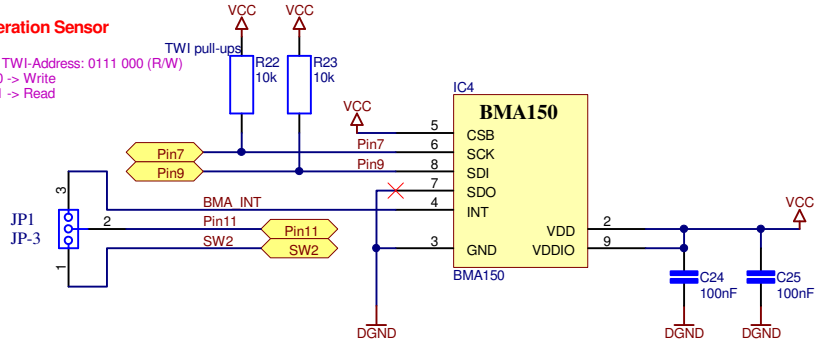
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFnode-1TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.101.02	Document File Name Interfaces.SchDoc		
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:44	DrawnBy: NOS/APA	Sheet 5 of 6	Rev: 2.0

4M Bit Flash Memory



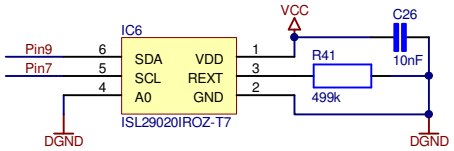
Acceleration Sensor

Internal TWI-Address: 0111 000 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



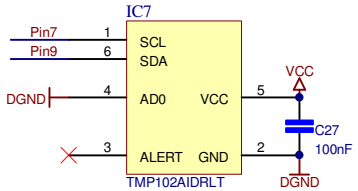
Ambient Light Sensor

Internal TWI-Address: 1000 100 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read

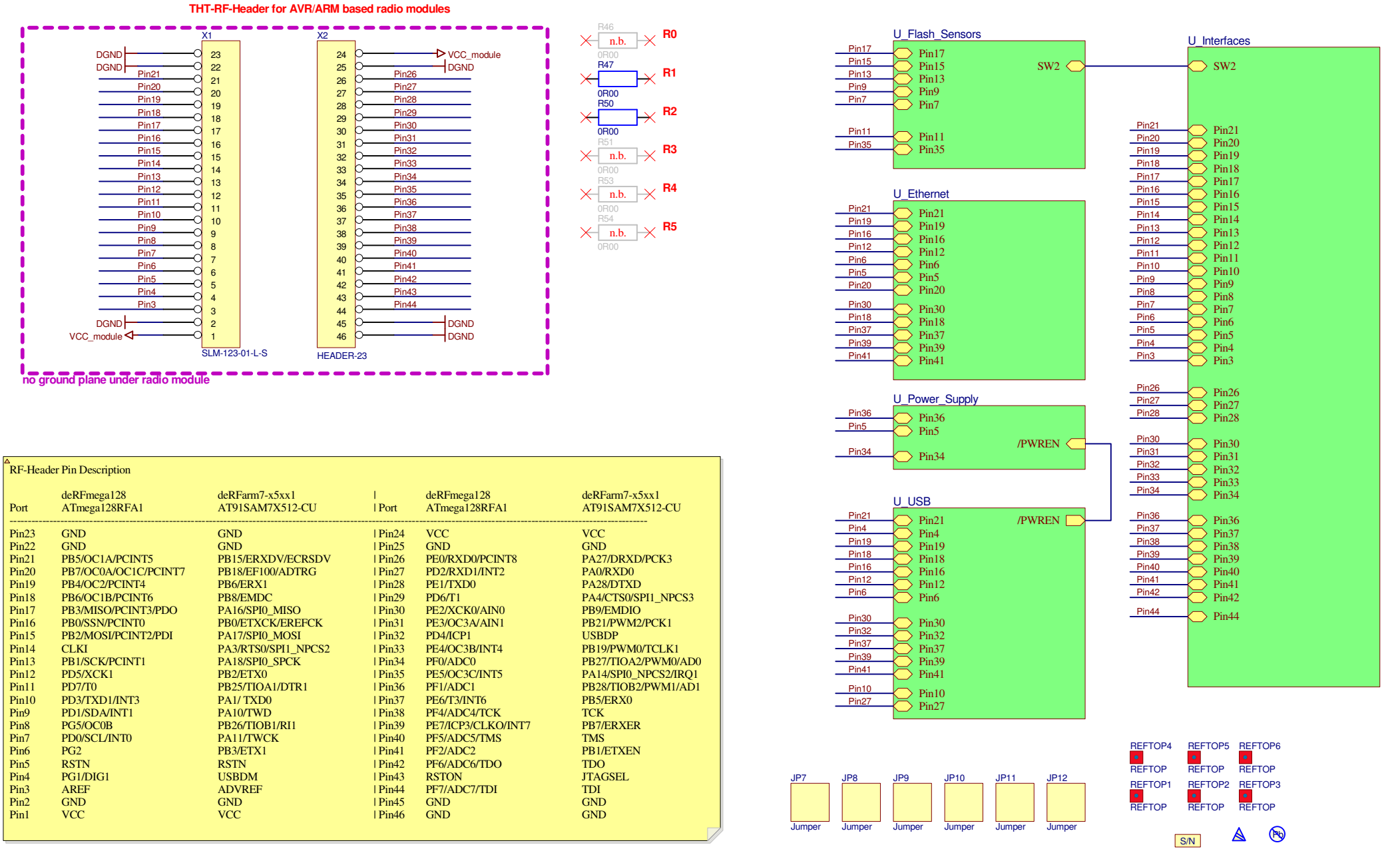


Temperature Sensor

Internal TWI-Address: 1001 000 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title **deRFnode-1TNP2-00N00**			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.101.02	Document File Name Flash_Sensors.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:13:45	DrawnBy: NOS/APA	Sheet 6 of 6 / Rev: 2.0

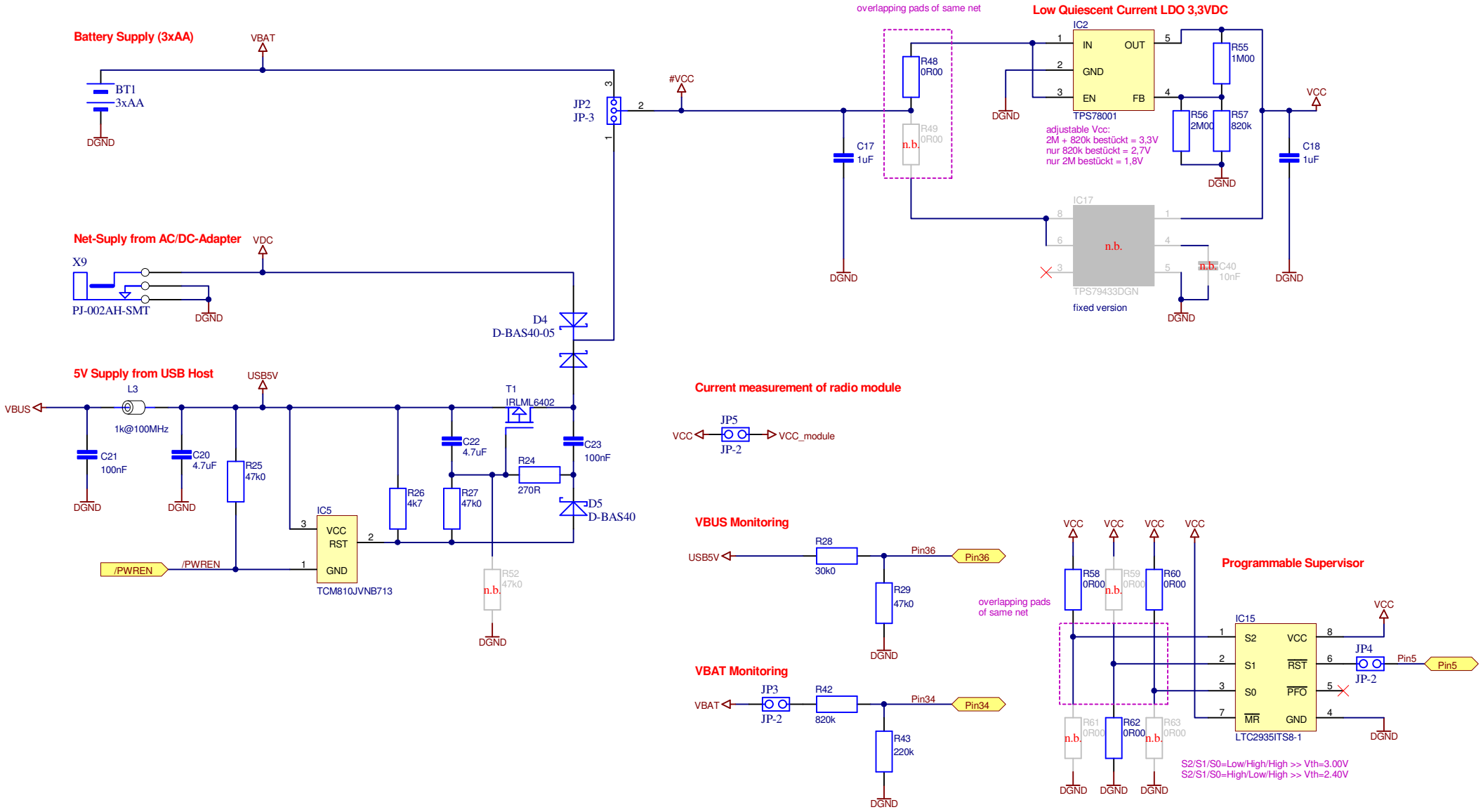


RF-Header Pin Description

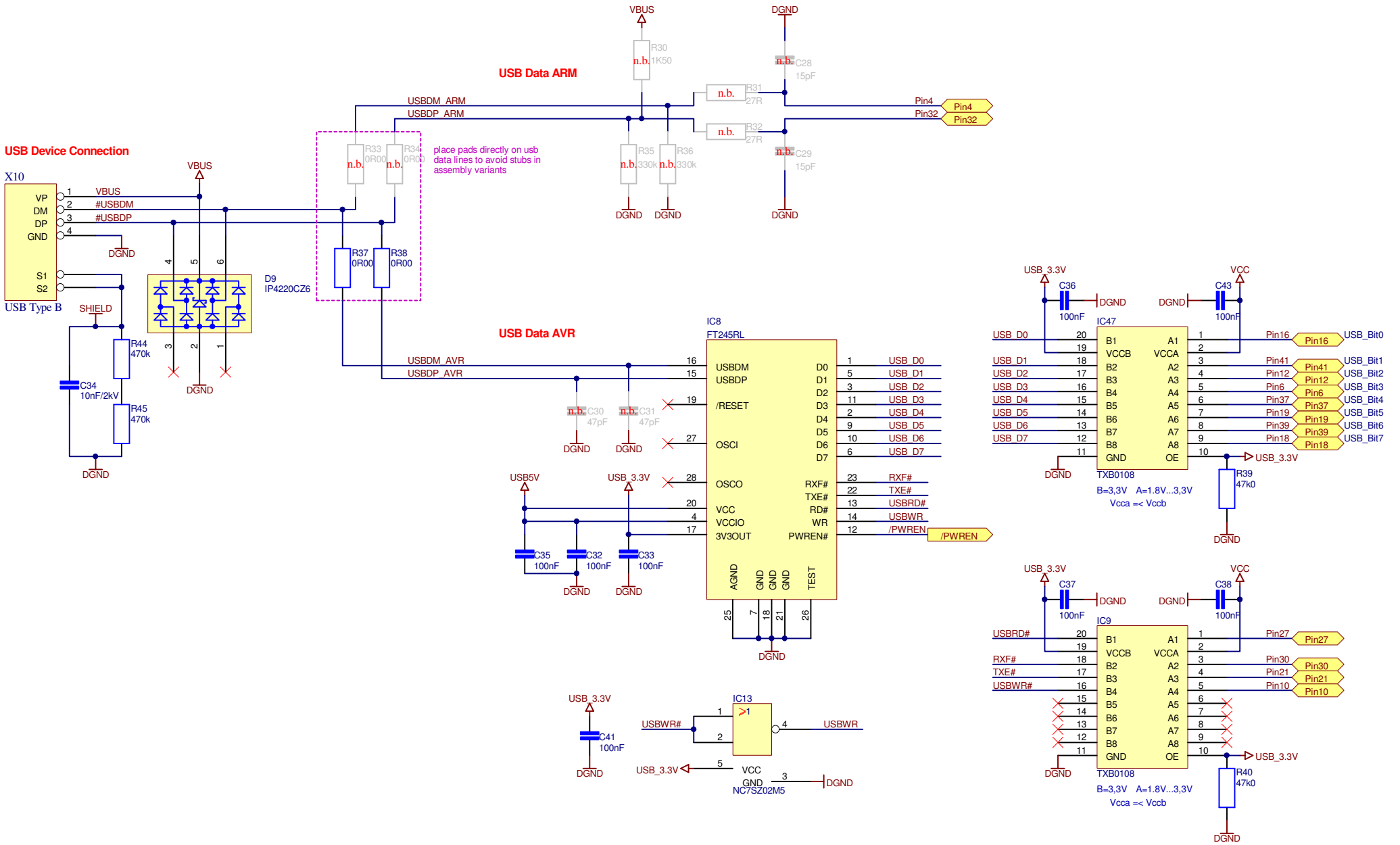
Port	deRFmega128 ATmega128RFA1	deRFarm7-x5xx1 AT91SAM7X512-CU	Port	deRFmega128 ATmega128RFA1	deRFarm7-x5xx1 AT91SAM7X512-CU
Pin23	GND	GND	Pin24	VCC	VCC
Pin22	GND	GND	Pin25	GND	GND
Pin21	PB5/OC1A/PCINT5	PB15/ERXDV/ECRSDV	Pin26	PE0/RXD0/PCINT8	PA27/DRXD/PCK3
Pin20	PB7/OC0A/OC1C/PCINT7	PB18/EF100/ADTRG	Pin27	PD2/RXD1/INT2	PA0/RXD0
Pin19	PB4/OC2/PCINT4	PB6/ERX1	Pin28	PE1/TXD0	PA28/DTXD
Pin18	PB6/OC1B/PCINT6	PB8/EMDC	Pin29	PD6/T1	PA4/CTS0/SPI1_NPCS3
Pin17	PB3/MISO/PCINT3/PDO	PA16/SPI0_MISO	Pin30	PE2/XCK0/AIN0	PB9/EMDIO
Pin16	PB0/SSN/PCINT0	PB0/ETXCK/EREFCK	Pin31	PE3/OC3A/AIN1	PB21/PWM2/PCK1
Pin15	PB2/MOSI/PCINT2/PDI	PA17/SPI0_MOSI	Pin32	PD4/ICP1	USBDP
Pin14	CLKI	PA3/RTS0/SPI1_NPCS2	Pin33	PE4/OC3B/INT4	PB19/PWM0/TCLK1
Pin13	PB1/SCK/PCINT1	PA18/SPI0_SPCK	Pin34	PF0/ADC0	PB27/TIOA2/PWM0/AD0
Pin12	PD5/XCK1	PB2/ETX0	Pin35	PE5/OC3C/INT5	PA14/SPI0_NPCS2/IRQ1
Pin11	PD7/T0	PB25/TIOA1/DTR1	Pin36	PF1/ADC1	PB28/TIOB2/PWM1/AD1
Pin10	PD3/TXD1/INT3	PA1/ TXD0	Pin37	PE6/T3/INT6	PB5/ERX0
Pin9	PD1/SDA/INT1	PA10/TWD	Pin38	PF4/ADC4/TCK	TCK
Pin8	PG5/OC0B	PB26/TIOB1/RI1	Pin39	PE7/ICP3/CLKO/INT7	PB7/ERXER
Pin7	PD0/SCL/INT0	PA11/TWCK	Pin40	PF5/ADC5/TMS	TMS
Pin6	PG2	PB3/ETX1	Pin41	PF2/ADC2	PB1/ETXEN
Pin5	RSTN	RSTN	Pin42	PF6/ADC6/TDO	TDO
Pin4	PG1/DIG1	USBDM	Pin43	RSTON	JTAGSEL
Pin3	AREF	ADVREF	Pin44	PF7/ADC7/TDI	TDI
Pin2	GND	GND	Pin45	GND	GND
Pin1	VCC	VCC	Pin46	GND	GND

dresden elektronik ingenieurtechnik gmbh
D-01237 Dresden, Enno-Heidebroek-Str. 12
Telefon 0351 / 31 85 00
Telefax 0351 / 31 85 010

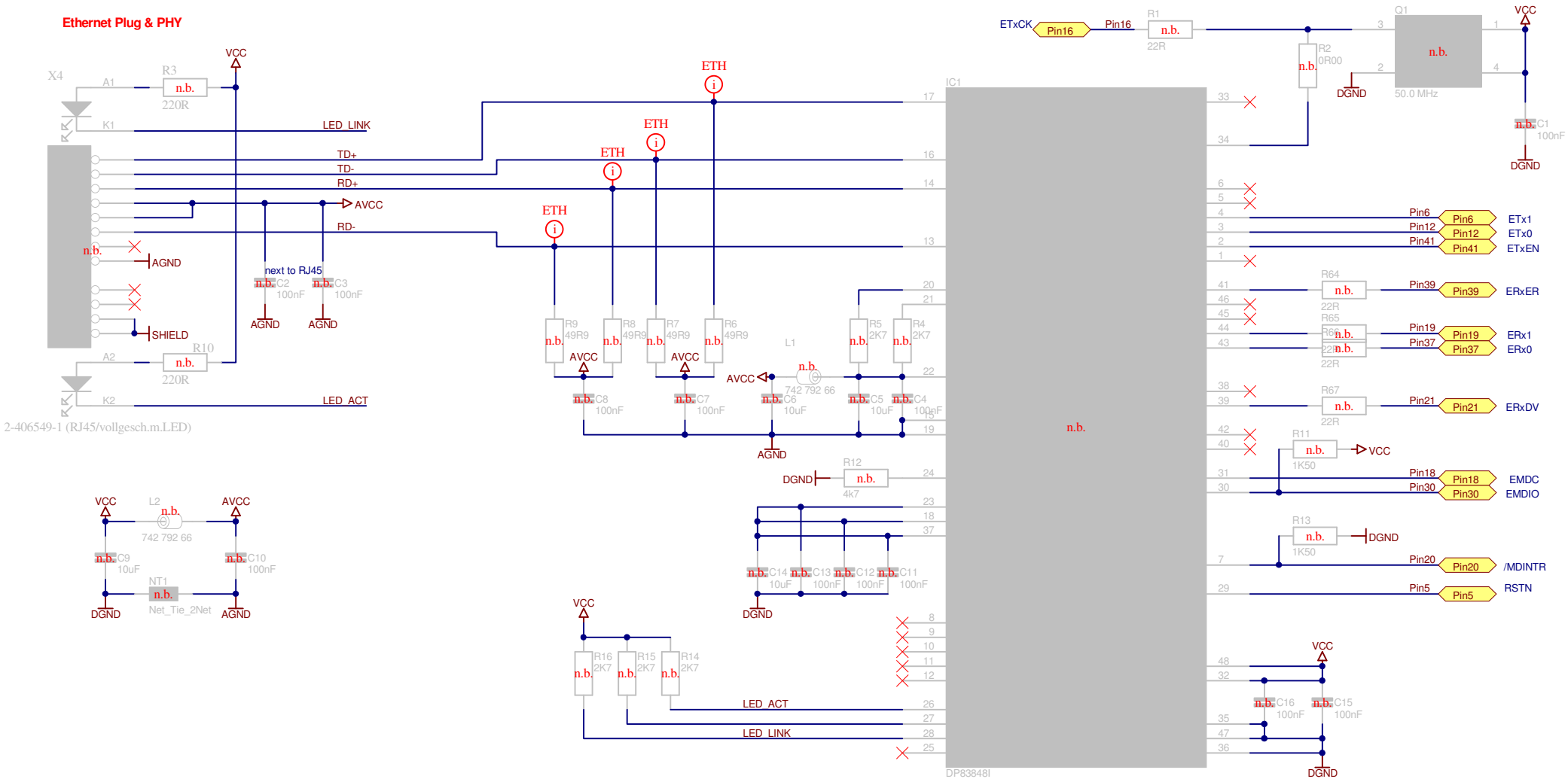
Title: deRFnode-2TNP2-00N00
Size: A4
Document Number: 5 646 16 02.105.02
Document File Name: Module_Connectors.SchDoc
Nr. | Änderung | Datum | Name
Date: 18.03.2011 | Time: 10:14:03 | DrawnBy: NOS/APA | Sheet 1 of 6 | Rev: 2.0



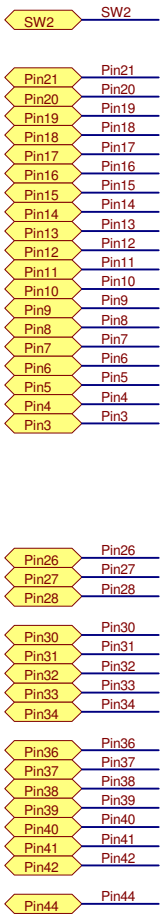
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title deRFnode-2TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.105.02	Document File Name Power_Supply.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:14:03	DrawnBy: NOS/APA	Sheet 2 of 6 Rev: 2.0



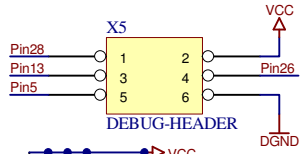
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFnode-2TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.105.02	Document File Name USB.SchDoc		
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:14:03	DrawnBy: NOS/APA	Sheet 3 of 6	Rev: 2.0

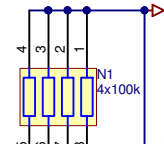


				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12				
				Title deRFnode-2TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010	
				Size A4	Document Number 5 646 16 02.105.02		Document File Name Ethernet.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011		Time: 10:14:03	DrawnBy: NOS/APA	Sheet 4 of 6 Rev: 2.0

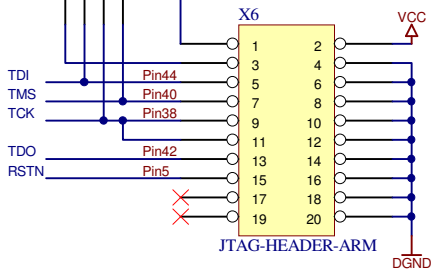


Debug-Header

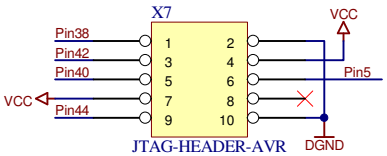




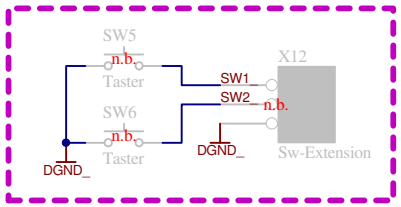
JTAG-Header for ARM



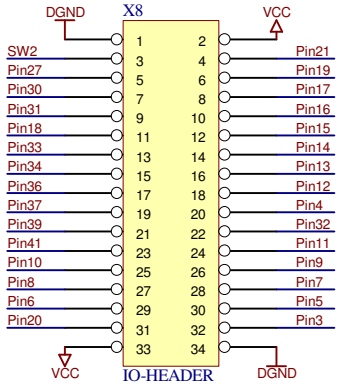
JTAG-Header for AVR



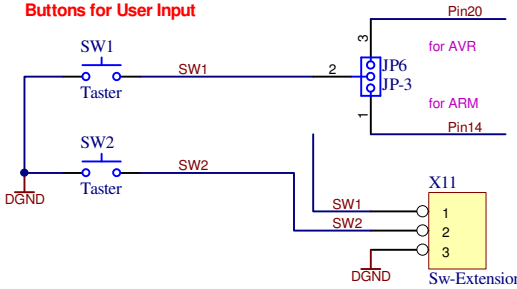
Separated PCB - only for case version



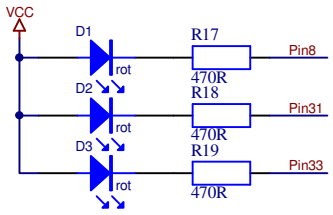
I/O-Header



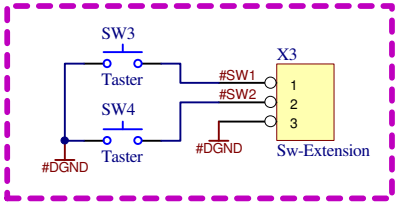
Buttons for User Input



LEDs for User Output

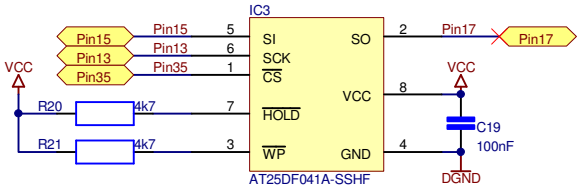


Separated PCB - only for case version



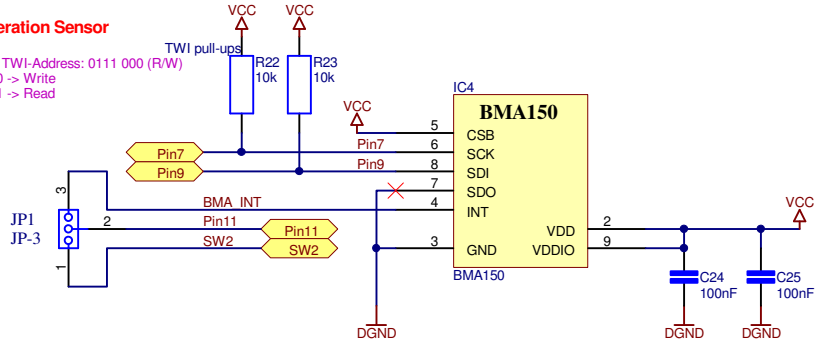
				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title deRFnode-2TNP2-00N00			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.105.02	Document File Name Interfaces.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:14:04	DrawnBy: NOS/APA	Sheet 5 of 6 / Rev: 2.0

4M Bit Flash Memory



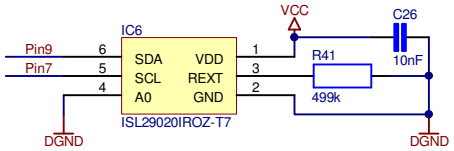
Acceleration Sensor

Internal TWI-Address: 0111 000 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



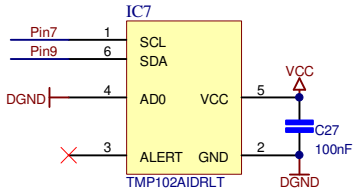
Ambient Light Sensor

Internal TWI-Address: 1000 100 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



Temperature Sensor

Internal TWI-Address: 1001 000 (R/W)
R/W = 0 -> Write
R/W = 1 -> Read



				dresden elektronik ingenieurtechnik gmbh D-01237 Dresden, Enno-Heidebroek-Str. 12			
				Title **deRFnode-2TNP2-00N00**			Telefon 0351 / 31 85 00 Telefax 0351 / 31 85 010
				Size A4	Document Number 5 646 16 02.105.02	Document File Name Flash_Sensors.SchDoc	
Nr.	Änderung	Datum	Name	Date: 18.03.2011	Time: 10:14:04	DrawnBy: NOS/APA	Sheet 6 of 6 / Rev: 2.0