Flamingo™ Interface Board for Atmel RF Modules

User Manual

Table of Contents

Section 1

Overview 1-1

1.1 Features 1-1
1.2 Supported RF Chipsets 1-1
1.3 Description 1-1
1.4 Overview 1-2
1.4.1 Modular System Architecture 1-3
1.5 Features and Capabilities of the Flamingo 1-4
1.5.1 General Features 1-4
1.5.2 Features in Combination with STK500 Starter Kit 1-4
1.5.3 Features as Stand-alone Unit in Combination with Battery-board 1-4
1.5.4 Software Environment 1-4

Section 2

Hardware Description 2-1

2.1 Flamingo™ 2-1
2.1.1 ATR2406 Interface (J11) 2-2
2.1.2 ATR86RF211 Interface (J30, J21, J1-J5) 2-3
2.1.3 Interfaces (X1, X2, J37) 2-5
2.1.4 Voltage Regulation Block 2-6
2.1.5 User Interface Block 2-7
2.2 Battery-board 2-9
2.2.1 ATmega16 2-9
2.2.2 Buffered Oscillator 2-9
2.2.3 Voltage Supply Block 2-9

Section 3

Software Libraries for Accessing the Atmel Flamingo™ User Interface via TWI-bus 3-1

3.1 Introduction 3-1
3.2 Software Package 3-1
3.3 How to Access the User Interface from the STK Side 3-2
3.4 Library-functions 3-5
3.4.1 Flamingo_ui.h 3-5
3.4.2 Switching the LEDs 3-5
3.4.3 Printing to LCD 3-5

3.4.4 Sleepmodes and External Interrupt Control 3-5
3.4.5 A/D Conversion 3-5
3.4.6 LED.....3-6
3.4.7 Printing to LCD 3-6
3.4.8 Sleepmodes and External Interrupt Control 3-6
3.4.9 A/D Conversion 3-6
3.4.10 General TWI Functions.....3-7
3.4.11 Interrupt Service Routines 3-7

Section 4
Schematics of Flamingo™ Board 4-1

Section 5
Connectors for RF Modules..... 5-1

Section 6
Abbreviations 6-1

Section 1

Overview

1.1 Features

- **Modular Starter System and Development Kit for Various RF Modules**
- **Easy Plug-and-play via Expansion-slot to AVR® Starter Kit STK®500**
- **Demonstration Capabilities and Stand-alone Operation with Battery-board**
- **Selectable Supply-voltage for RF Modules (2.4 V, 3.0 V, 3.4 V, 4.5 V, 5.0 V)**
- **Extensive User Interface Section, Accessible via Two-wire-bus**
- **Common Reference Clock for AVR and RF Module**

1.2 Supported RF Chipsets

Transceiver ICs	ATR2806, ATR2406, AT86RF210, AT86RF21x
Transmitter ICs	ATA5750, ATA5753, ATA5754, ATA5756, ATA5757
Receiver ICs	ATA5743, ATA5760, ATA5761

1.3 Description

The Flamingo™ is a starter system and development kit for various Atmel RF chipsets. It gives access to the complete range of AVR Flash microcontrollers via easy plug-and-play to the STK500 starter kit through the expansion slot. The system has a modular approach and hosts a set of interface connectors for different RF modules. The RF modules are available separately. RF modules are currently available for low power, low data rate automotive applications such as Remote Keyless Entry (RKE) and Tire Pressure Monitoring (TPM) but also for general purpose ISM applications such as high-speed data transmission on 2.4 GHz.

The Flamingo allows designers a quick start to develop RF-based solutions focusing on developing software code rather than obtaining RF related issues such as RF board layout and RF fine tuning. Combined with features for rapid prototyping and testing, provided by the STK500, the design cycle for new designs can be dramatically reduced.

In addition, the Flamingo also offers an ideal platform for demonstration capability. By replacing the STK500 with a battery-board, the complete system – consisting of RF module, interface board and battery-board – can be operated independently from a PC or a power supply. So this can be used either for demonstration, evaluation purposes or doing rapid prototypes.

1.4 Overview

Figure 1-1 shows the Flamingo board with descriptions of the various embedded components.

Figure 1-1. Top View of Flamingo

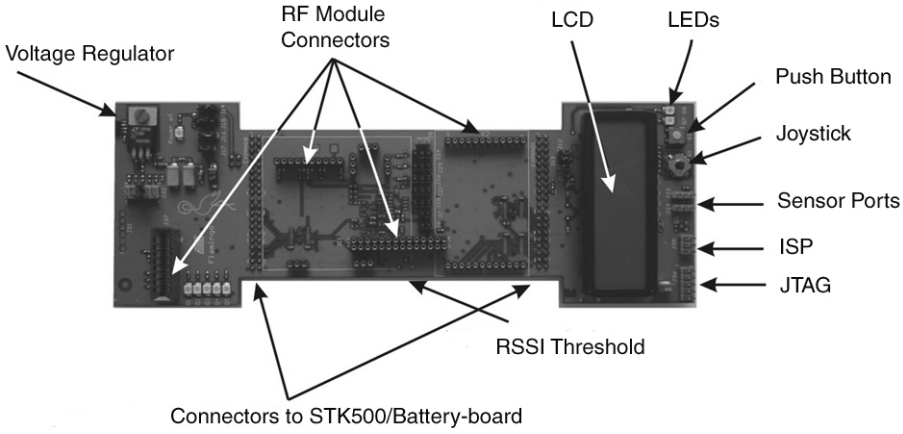


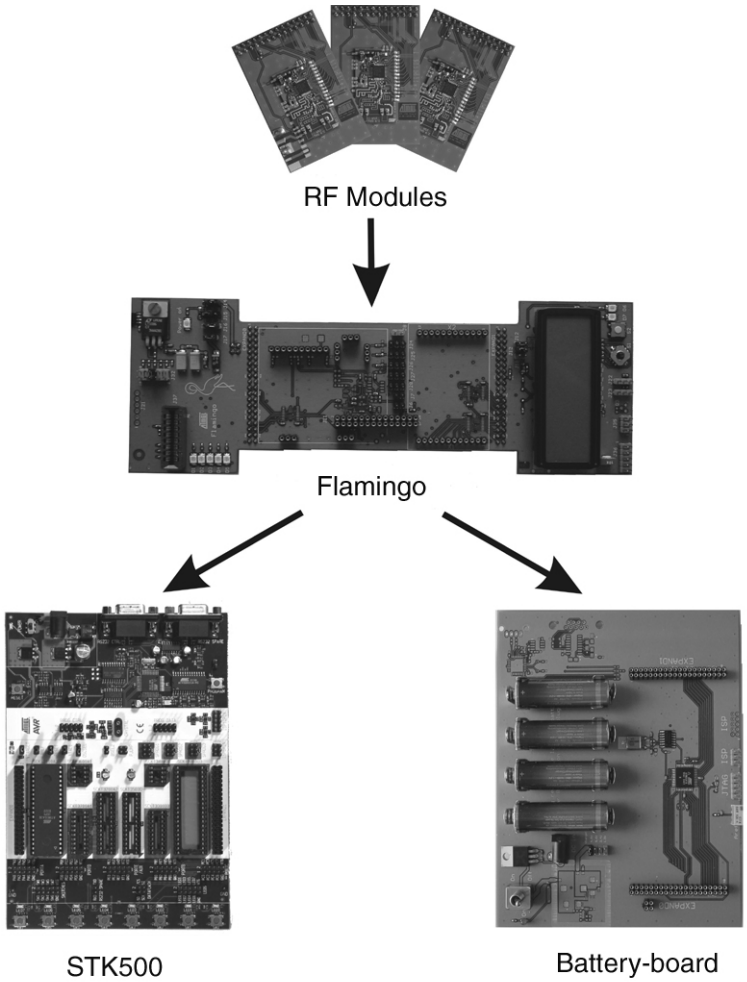
Table 1-1. Items of the Flamingo Evaluation Kit

Item	Number of Items	Comment
Flamingo evaluation board	1	Standard
User manual	1	Standard
Battery-board	1	Optional available, not included
RF module	-	To be ordered separately, not included
STK500 starter kit	-	Optional available, not included

1.4.1 Modular System Architecture

Figure 1-2 demonstrates the modular system approach of the Flamingo.

Figure 1-2. Modular System Concepts



1.5 Features and Capabilities of the Flamingo

1.5.1 General Features

- Flexibility of using the RF modules in different modes, selectable via jumpers
- Individually selectable supply voltage for the RF modules (i.e., 2.4 V, 3.0 V, 3.4 V, 4.5 V, 5.0 V)
- Five status LEDs directly accessible via the host controller, alternatively, ports can be used as GPIOs
- User Interface section, accessible via a 2-wire-bus, with the following capabilities:
  - 4 × 16 char dot-matrix display for enhanced demonstration ability
  - Joystick for menu control
  - One push button
  - Two status LEDs accessible via TWI
  - Three ports (A/D capable) on pinheads available for auxiliary sensors (temperature, light intensity, 1-wire bus devices, etc.)
  - Buzzer
  - 32 kHz XTAL for sleep mode (optional)
  - Four additional auxiliary GPIOs (one IRQ, one A/D, two general purpose digital I/Os)
  - JTAG and ISP headers on board for programming the User Interface processor
- Integrated high bit rate data-slicer for digitizing analog RX_DATA signals (optional)
- Common reference clock for AVR and RF module supplied either by STK500 or from battery-board
- Working in combination with STK500 (for evaluation/development, connectable via expansion connector) or as a stand-alone unit (for demonstration purposes) together with battery-board

1.5.2 Features in Combination with STK500 Starter Kit

- Flexibility in using different AVRs according to the application requirements, i.e., for high data rate, low power consumption, etc.
- Status display via STK500 LEDs or Flamingo User Interface

1.5.3 Features as Stand-alone Unit in Combination with Battery-board

- Stand-alone unit together with battery-board includes its own host processor for driving the RF modules
- Host processor AVR on battery-board is ATmega16
- Commands/control via joystick/push button on User Interface
- Integrated JTAG/ISP adapters
- Onboard Status LED for e.g., TX_ON signal

1.5.4 Software Environment

- Atmel provides free of charge, a CodeVision compatible project with complete source code for the reference design evaluation board demo applications, as well as the User Interface section

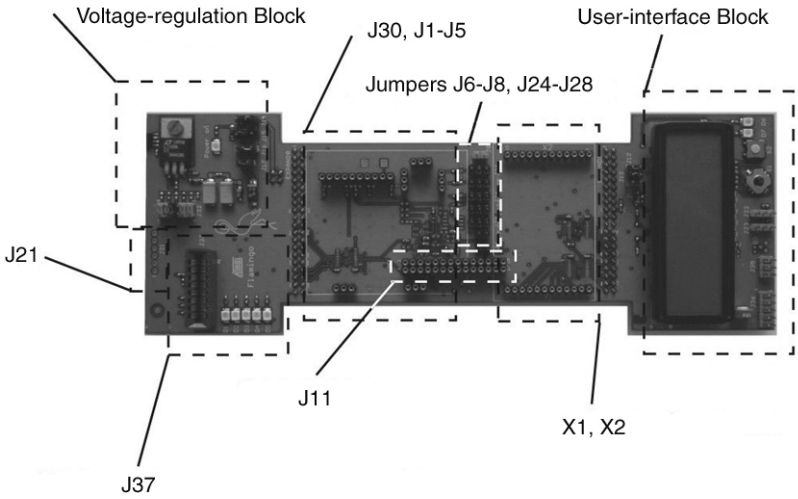
Section 2

Hardware Description

2.1 Flamingo™

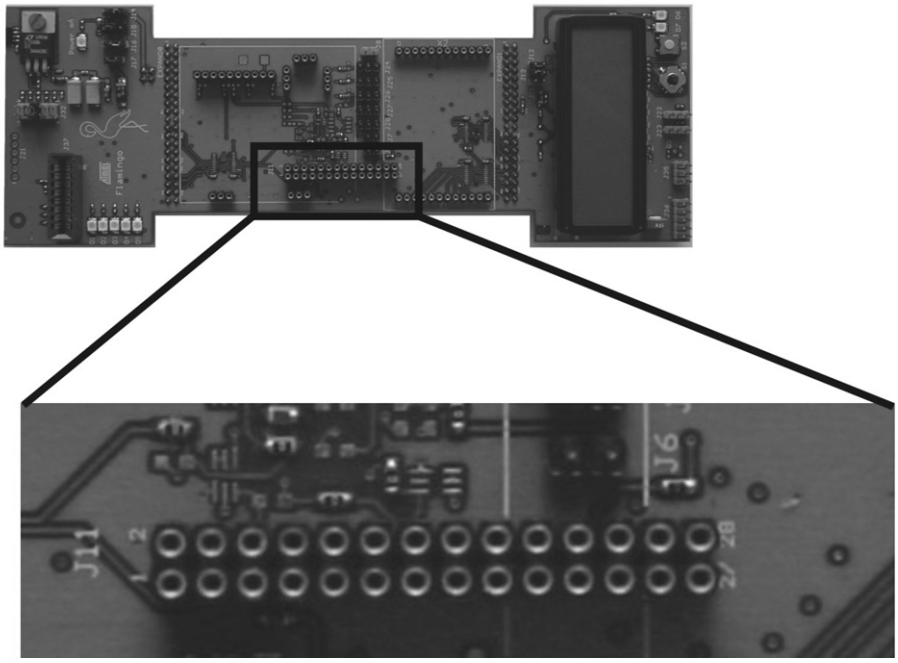
To facilitate the use of Flamingo with a wide range of Atmel's RF products, different connectors and interfaces are included on the board. Through these interfaces, each RF module is connected to the underlying microcontroller unit on the STK®500 or battery-board.

Figure 2-1. Connectors and Interfaces



2.1.1 ATR2406 Interface (J11)

Figure 2-2. Interface Enlarged (J11)



Through jumper J11 the ATR2406 is connected to the Flamingo and the microcontroller of the battery-board or the STK500. The pinout of the interface is shown in Figure 2-3.

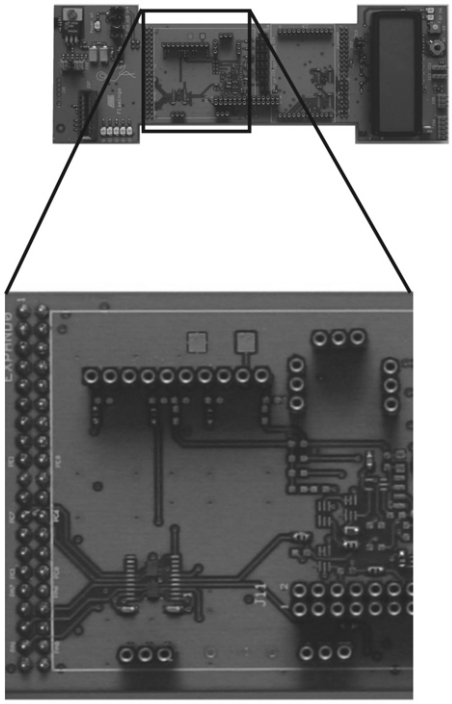
Figure 2-3. J11 Pinout

J11

V_RF	1	2	V_RF
TX_ON (PA4_S)	3	4	ANT DIV. (PA5)
TXDATA (PD1_S)	5	6	PU_PLL
RESERVED	7	8	3-WIRE END (PD7_S)
ICP_SW (PD2_S)	9	10	3-WIRE DTA (PD6_S)
3-WIRE CLK (PD5_S)	11	12	OLE (PB1_S)
PU_RX/TX	13	14	PU_REG
RX_ON (PA3_S)	15	16	LD
GND	17	18	GND
GND	19	20	RX_DATA
RSSI	21	22	GND
GND	23	24	DATA SLIC. (PA1)
REF CLK	25	26	GND
GND	27	28	GND

2.1.2 ATR86RF211 Interface (J30, J21, J1-J5)

Figure 2-4. Interface Enlarged (J30, J21, J1-J5)



The connectors of the ATR86RF211 Interface block, shown in Figure 2-4, are used to connect the ATR86RF211 modules. Figure 2-5 on page 4 gives detailed information about the connector's pinout. The jumpers next to the connectors are described in Table 2-1 on page 4.

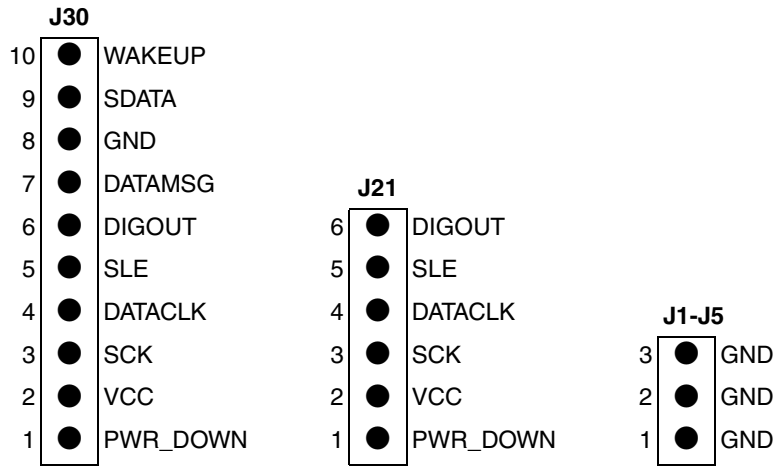
The RF module can be driven in three different modes. In USART Mode the RF module exchanges data with the microcontroller via its USART Serial Interface. In SPI mode the 3-wire serial interface is used. However, if the microcontroller plugged into the STK500 does not have SPI or USART interfaces the data exchange can be carried out by the direct manipulation of two pins in general purpose mode.The jumpers listed in Table 2-1 on page 4 allow the desired mode to be chosen. For details on working with ATR86RF211 and similar modules please consult the appropriate datasheets.

Table 2-1. Jumpers

Jumper	Setting	Function
USART		
J6	Connect	RxD of the MCU's USART is linked to the Serial Input of the RF211 (DataMSG)
	Disconnect	When USART is not used
J7	Connect	TxD of the MCU's USART is linked to the Serial Input of the RF211 (DataMSG)
	Disconnect	When USART is not used
SPI		
J24	1-2	SCK signal is taken from personal computer
	2-3	SCK signal is taken from PINB 1 of the MCU
J25	1-2	SLE signal is taken from personal computer
	2-3	SLE signal is taken from PINB 1 of the MCU
J26	1-2	SDATA signal is taken from PC
	2-3	SDATA signal is taken from PB1 of the MCU
General Purpose/SPI		
J27	1-2	Sets SPI Mode for exchanging data between MCU and RF module
	2-3	General purpose mode: the RF module triggers the INT1 pin as CLK signal
J28	1-2	Sets SPI Mode for exchanging data between MCU and RF module
	2-3	General purpose mode: the MCU can sample the data according to CLK

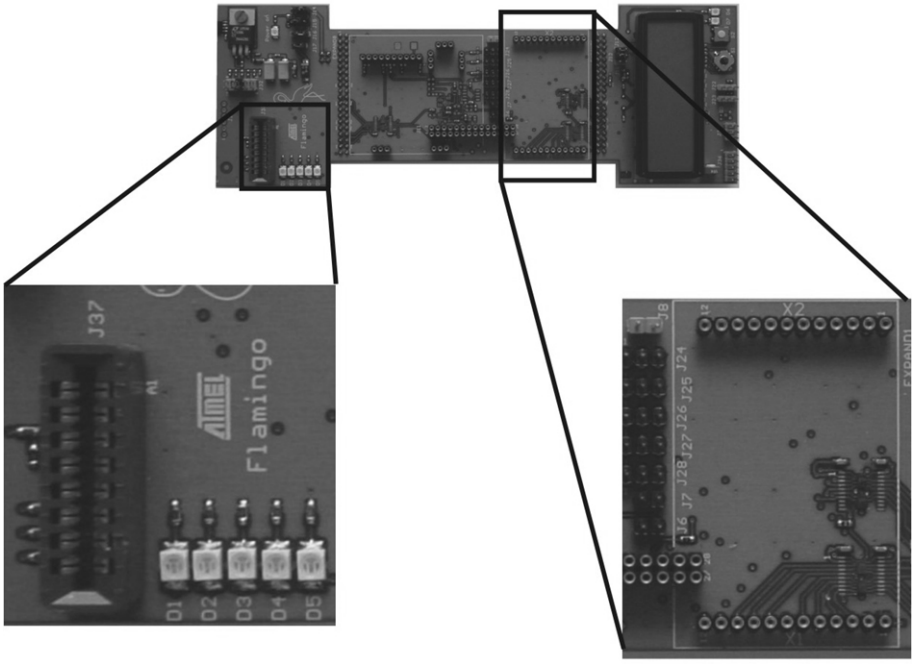
Figure 2-5 depicts the pinout of all connectors that will be used when working with the ATR86RF211 modules. The ATR86RF211 modules are plugged into J30. If a personal computer or another external device is used to exchange data via the SPI interface, this external device will be connected through J21. The connectors J1 to J5 are all connected to GND.

Figure 2-5. Pinout Connectors (J30, J21, J1-J5)



2.1.3 Interfaces (X1, X2, J37)

Figure 2-6. Interface Enlarged (X1, X2, J37)



When working with devices of the ATAB57xx series there are usually two modules involved. One module carries the transmitter chip and the other one carries the receiver. The receiver is plugged into the sockets X1 and X2 mounted on the Flamingo. Enlarged pictures are shown in the lower half of Figure 2-6. The pinout is depicted in Figure 2-7 on page 6.

The transmitter device is programmed by the STK500's microcontroller. Therefore it will be plugged into the socket J37 that is mounted in the area enlarged in the left part of Figure 2-6. To program the transmitter special software was created. The datasheets of the ATAB57xx will give detailed information on this subject.

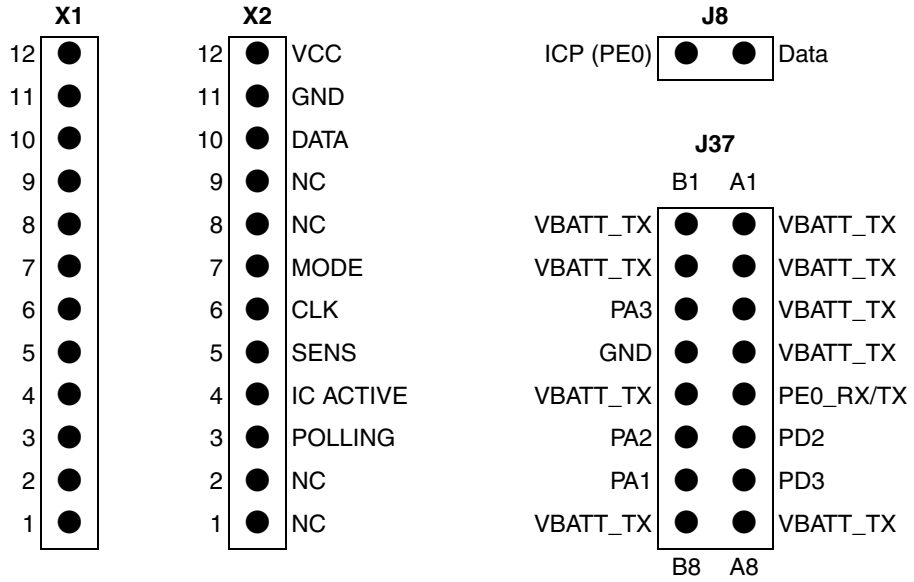
The LEDs mounted next to J37 are used to indicate different states. Table 2-2 gives information about the function of each LED.

Table 2-2. ATAB57xx LED Description

LED	Function
D1	Constant glowing - AVR® is ready to receive data Flashing - data is being received
D2	Test word incorrect, MCU limits ok
D3	MCU limits out of selected border
D4	Test word ok and MCU limits ok
D5	Communication via RS-232 active

Note: Jumper J8 (mounted next to port X2) has to be set when the AVR microcontroller that is plugged into the STK500 is capable of recognizing interrupts on its pin PE0 (PE0 = ICP), such as the ATmega8515. For example, it has to be disconnected when using the ATmega8535.

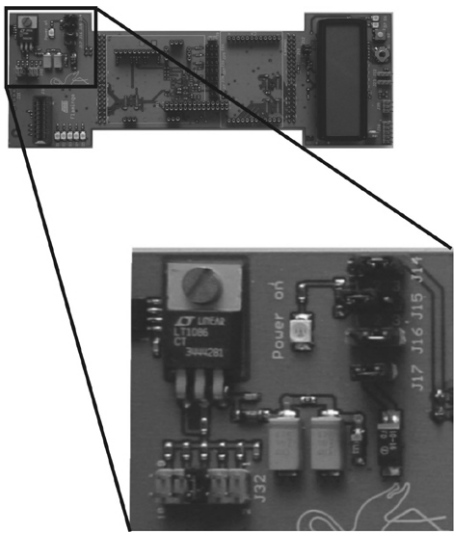
Figure 2-7. Pinout of Sockets X1, X2 and J37



Current status: X1 is not connected
All pins are reserved for future use

2.1.4 Voltage Regulation Block

Figure 2-8. Voltage Regulation Block Enlarged



There are two voltage regulators in the voltage regulation block. One regulator (7805 - details are given in section 4 “Schematics of the Flamingo Board” keeps the supply voltage provided by the battery-board or the STK500 at a constant level of 5 V. The jumper J15 can be set to bypass this regulator if it is not needed. For details refer to Table 2-3 on page 7.

The other regulator (LT1086) reduces the supply voltage to the appropriate level for each RF module V_{RF}. Using jumper J32, five different voltage levels can be chosen. If the RF module does not need a regulated supply voltage J16 can be used to bypass the regulator.

J17 offers the possibility to insert a current meter for measuring the RF module’s input current. During normal operation J17 should be connected otherwise the RF module is disconnected from the power supply.

Figure 2-9. Pinout of Voltage Regulation Jumpers J32, J14-J17

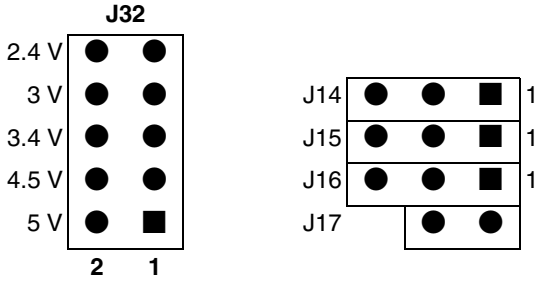


Table 2-3. Jumper Settings for Supply Voltage Modes

Jumper	Setting	Function
J14	1-2	Reserved
	2-3	Supply voltage taken from STK500 or battery-board
J15	1-2	5 V voltage regulator bypass
	2-3	Enable 5 V regulator
J16	1-2	V_{RF} regulator bypass
	2-3	Enable V_{RF} regulator
J17	Open	Open circuit for measuring I_{RF}
	Closed	V_{RF} is applied to RF module

2.1.5 User Interface Block

The User Interface devices are mounted in this area of the Flamingo as well as the ATmega16 and its connectors for the programming devices. The joystick and the push button are connected to one of the external interrupt pins of the ATmega16. This provides a flexible way to deal with user input and to implement menu functions for convincing demonstration strategies. There is an extensive software library with source code written in C included in this package. For more details on the software and a quick access to the User Interface refer to section 3 “Software Libraries for Accessing the Atmel Flamingo’s User Interface via TWI-bus”.

The ATmega16 can be programmed with the JTAG ICE or a serial programming device such as the STK500 or AVR ISP, therefore J35 and J36 have been mounted on the Flamingo. The serial programming devices have to be connected with the 6-wire cable to the ISP port J35 (Figure 2-10 on page 8) and the JTAG ICE has to be connected to J36.

When using analog sensors, or to measure other analog output values, the ports J22 and J23 can be used. They are linked to the ATmega16's internal AD-converter channels. The datasheet of the ATmega16 will give further details on using its AD-converters.

The small potentiometer next to J36 is used to adjust the contrast level of the 20x4 LCD.

Figure 2-10. User Interface Devices

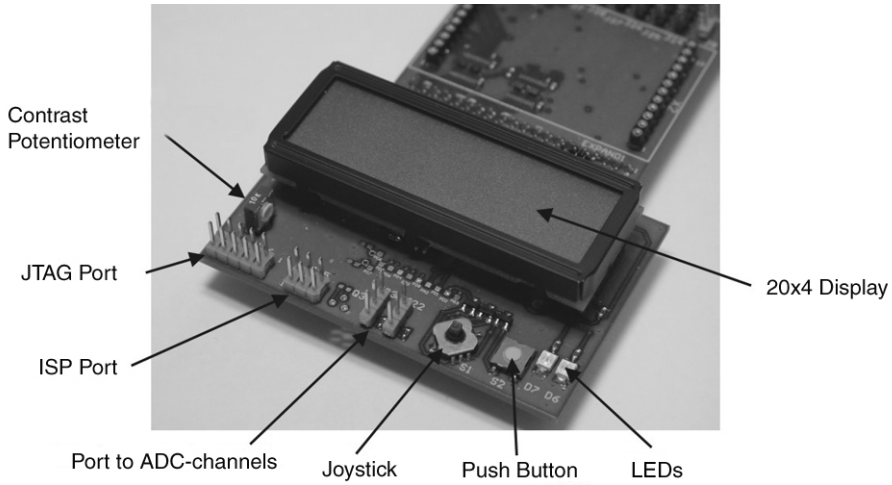
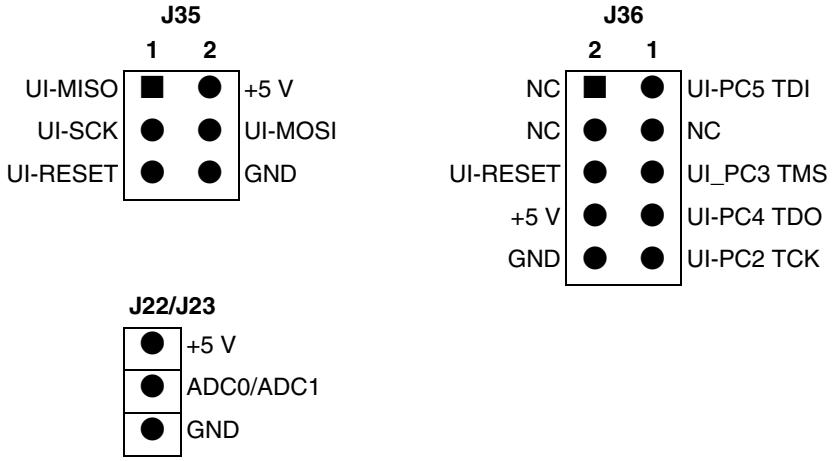


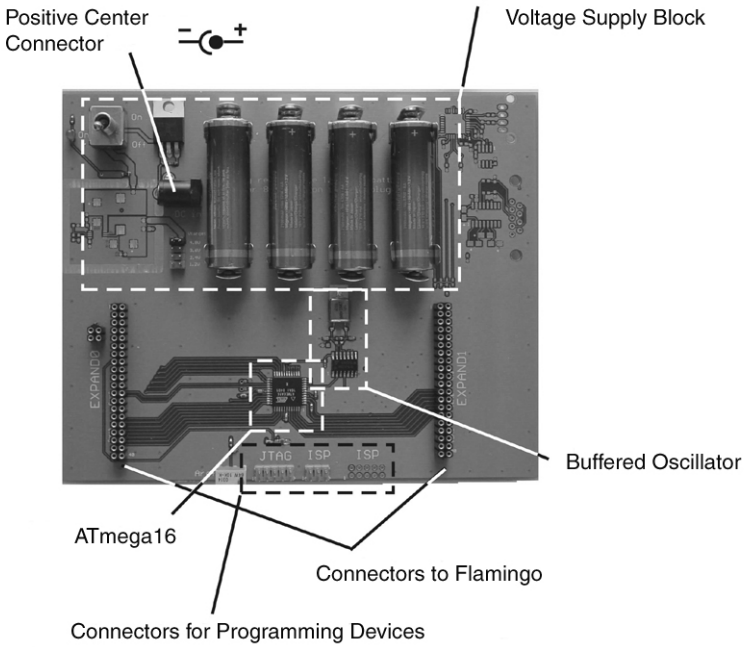
Figure 2-11. Pinout of J22/J23, J35 and J36 Located at User Interface



2.2 Battery-board

The intention of the battery-board is for demonstrating real “wireless” - without the need for an external power supply or personal computer as it is needed, i.e. for the STK500. So in combination with the Flamingo Interface Board and its user interface a complete stand-alone application can be demonstrated.

Figure 2-12. Battery-board Overview



2.2.1 ATmega16

Figure 2-12 depicts the hardware devices mounted on the battery-board. The microcontroller unit is a ATmega16 with 16 Kbyte Flash Memory. This MCU can be programmed with a serial programming device such as the STK500 or with the JTAG ICE. Therefore the programming device has to be connected to the assigned connectors. The serial programming device has to be connected to the socket named ISP while the JTAG is connected via the equally named socket.

2.2.2 Buffered Oscillator

A buffered oscillator is used to avoid loading effects in case if different types of RF modules are going to be used.

2.2.3 Voltage Supply Block

The battery-board offers two ways to provide supply voltage to the development kit. For stand-alone operations four rechargeable 1.2 V batteries are recommended. It is also possible to connect an external supply unit (8 ... 12 V DC) with the positive center connector to the external power plug. This plug is mounted in such a way that either the batteries can be used or the external supply unit, but not both at the same time.

WARNING: Using four 1.5 V batteries can cause damage to the ATmega16 because this MCU typically needs a supply voltage in the range from 4.5 V to 5.5 V.

Section 3

Software Libraries for Accessing the Atmel Flamingo™ User Interface via TWI-bus

3.1 Introduction

When developing RF-applications with the Atmel STK®500 and the Atmel Flamingo it is very useful to operate the Flamingo's UIDs from the STK500's MCU via the TWI bus. The possibility to display important data and changing configuration modes according to user input is fundamental for fast prototyping and a quick start of the development process. Therefore the software libraries *FLAMINGO_UI* and *STK_UI* have been created. These libraries provide several functions to establish a bi-directional exchange of information between two microcontrollers connected through their Two-Wire-Interfaces. The files *FLAMINGO.C* and *STK.C* contain source-code examples to illustrate the usage of some of the library-functions and how the information exchange between the MCUs can be arranged using interrupt-based communication via TWI. At the end of this document there is a list of all these functions with short descriptions. All the programs have been created by using the CodeVision AVR C Compiler.

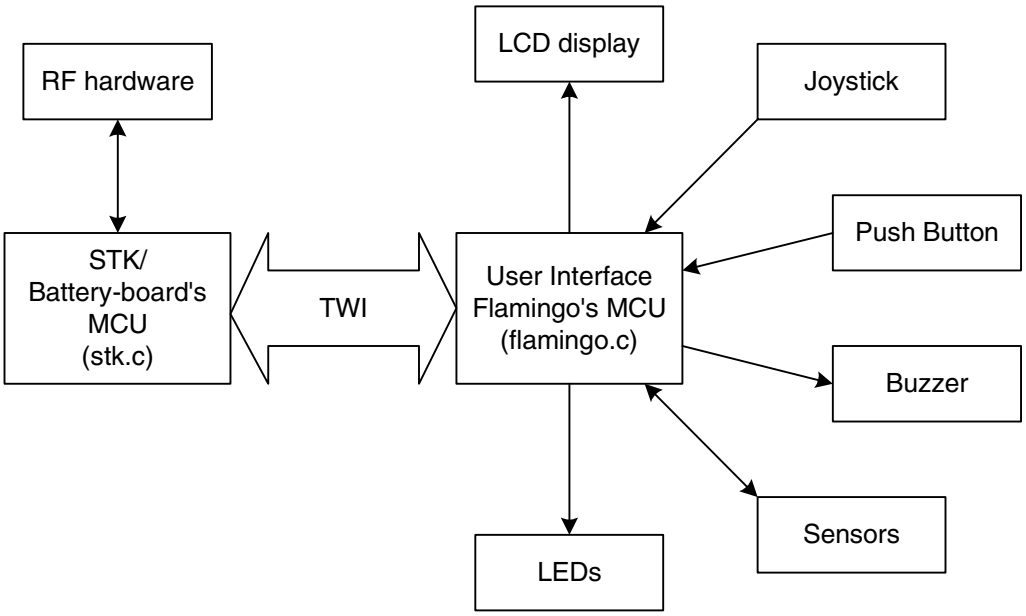
For more details on TWI serial communication please read the section "Two-wire Serial Interface" in the datasheet of the ATmega16.

3.2 Software Package

STK_UI.H/STK_UI.LIB	is the software library for programming the STK500 or battery-board. The libraries are optimized for the CodeVision AVR compiler. In STK_UI.H all the function headers are listed. STK_UI.LIB contains the implementation of the functions as source code.
STK.C	is the example program for the STK500 that shows how to use the functions listed in STK_UI.H and how to implement proper listening on the TWI bus for incoming user actions, such as pressing the push button or moving the joystick.
FLAMINGO_UI.H/ FLAMINGO_UI.LIB	is the library which contains the functions that drive the Flamingo's UIDs such as LCD, LED, etc.

FLAMINGO.C is the program that is already installed on the Flamingo's MCU. This program listens for incoming data on the TWI bus. It will analyze the commands and take the appropriate action, e.g. switch on/off the LEDs, place characters on the LCD or start an A/D conversion.

Figure 3-1. Schematic Overview of the User-interface's Data-flow



3.3 How to Access the User Interface from the STK Side

To provide access to the Flamingo's user-interface devices via the TWI bus, the file *FLAMINGO.C* has to be compiled and stored into the program memory of the Flamingo's microcontroller. This program causes the microcontroller to wait for incoming data on the TWI bus and to send data if the joystick is moved or the push button is pressed.

The incoming byte sequences are analyzed and if a valid command, such as "print characters to the LCD" or "switch on/off the LEDs", is recognized the user-interface devices are operated accordingly. The byte sequences that represent valid commands are described in Table 3-1 on page 4.

To operate the UI devices from the program running on the STK's microcontroller the functions implemented in *STK_UI.H* can be used. These functions can easily be called up including the library *STK_UI.H* and performing several steps for initialization.

First of all, the device's slave address has to be set because TWI protocol definitions require every component to have its unique bus address. By default configuration the Flamingo's MCU will respond to TWI messages delivered to the bus address 0x02 and will send its own messages to the address 0x04. Consequently the STK's MCU has to be initialized with 0x04 as its own slave address by calling the function twiInit (0x04,0x00). The second parameter of twiInit() is the bit rate denominator. Increasing this value will reduce the transmission frequency between the microcontrollers according to the expression in the datasheet of the ATmega16, section "Bit Rate Generator Unit, Two-wire Serial Interface."

The remote address, to which the commands will be send to, has to be assigned to 0x02 with setRemoteAddress(0x02).

Enabling recognition of joystick motion or push button usage within the STK's program the external interrupt (EXT INT0) flag of the Flamingo's MCU has to be set with twiSetExternalInterrupt(0x01).

Every valid TWI transmission will cause the execution of the TWI interrupt service routine, which will automatically be integrated in the software by including *STK_UI.H*. This implies that the global interrupt flag has to be set. After these steps the functions listed in the section *STK_UI.H* at the end of this text can be used to control the user-interface elements over the TWI bus.

To recognize joystick or push button events the STK's MCU has to receive and analyze incoming TWI data to extract information about the status of these devices. A source-code example for dealing with such TWI messages is delivered in the file *STK.C*. More details concerning these messages are described in Table 3-1 on page 4. Generally the first byte of the received data indicates what kind of information arrived during the last transmission. The following bytes will then give detailed information about the event that took place.

For example, when the first byte equals the transmission code for a joystick event the following byte will give information about the direction the joystick was moved to. For details on these transmission codes and how many bytes have to be analyzed in each case see Table 3-1 on page 4.

To analyze the incoming TWI data the return value of *newBytesReceived()* can be used to get the number of bytes that recently have been stored in the library's receive buffer. It will return the correct number of bytes just for one time, because it sets the library-variable *bytesReceived* to zero when it is called and new data is located in the buffer.

This mechanism was implemented to let the microcontroller return from its TWI interrupt service routine before executing any other functions. When executing the interrupt routine all interrupts of the same or lower priority are blocked as long as all the instructions are executed. This makes it impossible to initiate another TWI transaction while still analyzing the received information within the TWI interrupt service routine.

In the example *STK.C* the return value of *newBytesReceived()* is used to indicate if new data has been received and to execute directly the function *processBytes()* which extracts the information contained in the byte sequence and performs the appropriate action. The function stores the received bytes in a character array where they generally are accessible to the programmer. Then the first in the array byte, representing the kind of information, is used as an input parameter in a switch-case statement that branches to the proper instructions. This switch-case statement can be used to execute user-defined functions according to the recent joystick or push button event.

Table 3-1. Transmission Codes for Access to the User Interface

Transmission Code Byte 0	Library Definition	Direction	Number of Bytes	Information Bytes Related to Each Transmission Code			
				Byte 1	**Byte 2**	**Byte 3**	**Byte 4**
0x01	PUSHBUTTON	From Flamingo	2	PUSHBUTTON_PRESS (0x20)	-	-	-
0x02	JOYSTICK	From Flamingo	2	Directions: JOYSTICK_LEFT (0x01) JOYSTICK_UP (0x02) JOYSTICK_RIGHT (0x04) JOYSTICK_DOWN (0x08) JOYSTICK_ENTER (0x10)	-	-	-
0x03	LED	To Flamingo	2	LED_RED (0x01) LED_GREEN (0x02) LED_ALL_ON (0x03) LED_ALL_OFF (0x00)	-	-	-
0x04	LCD_CHAR	To Flamingo	3	Position	Character	-	-
0x05	LCD_CURSOR	To Flamingo	3	Position	Cursor	-	-
0x06	LCD_CLEAR	To Flamingo	1	-	-	-	-
0x07	LCD_DECIMAL	To Flamingo	4	Position	High byte	Low byte	-
0x08	LCD_STRING	To Flamingo	4-23	Position	Number of characters	First character	Second character
0x09	ANALOG_INPUT	To/from Flamingo	3 2	High byte AD-channel	Low byte	-	-
0x0A	STATUS_INT0	To Flamingo	2	Status: INT0_ON (0x01) INT0_OFF (0x00)	-	-	-
0x0B	SLEEP_MODE	To Flamingo	2	Sleep modes: SLEEP_MODE_IDLE (0x40) SLEEP_MODE_ADC (0x50) SLEEP_MODE_PDOWN (0x60) SLEEP_MODE_PSAVE (0x70) SLEEP_MODE_STANDBY (0xE0) SLEEP_MODE_EXT_STANDBY (0xF0)	-	-	-
0x0C	BUZZER	To Flamingo	5	Frequency high byte	Frequency low byte	Duration high byte	Duration low byte
0x0D	SET_REGISTER	To Flamingo	3	Register memory address	Value	-	-

3.4 Library-functions

3.4.1 Flamingo_ui.h

This library has to be included if the firmware functions are used to program the Flamingo's microcontroller.

3.4.2 Switching the LEDs

void outputLed(unsigned char illumination)
Switching on/off the green and red LED – setting the LSB of the parameter to 1 will switch on the red LED. Setting the second LSB of the parameter to 1 will switch on the green LED.

3.4.3 Printing to LCD

void outputLcdChar(unsigned char position, unsigned char character)
Prints an ASCII character to the LCD – the position is encoded, the bits 6 and 5 represent the y-position on the LCD, the bits 4 to 0 represent the x-position on the LCD, you can easily encode by "logical or" - combining the values of x and the value of y multiplicated by 0x20.
Example: x-position 10, y-position 3, is encoded encPos = 0x0A | 0x60 or

void ouputLcdCursor(unsigned char position, unsigned char cursor)
This function changes the position and appearance of the cursor. The parameter *position* is the encoded position, for details see the command outputLcdChar. The parameter *cursor* can be 0x0E for a blinking underscore or 0x0F for a blinking square.

*void outputLcdString(unsigned char position, unsigned char length, unsigned char *ptr)*
This function will print a whole string of characters addressed by the pointer ptr. The position is again encoded like the parameter of *outputLcdChar*. The parameter length is the number of characters.

void outputLcdDecimalInt(unsigned char position, int intValue)
This function prints a five digit integer in decimal format to the LCD. The position is encoded like the parameter of *outputLcdChar*.

3.4.4 Sleepmodes and External Interrupt Control

void setSleepMode(unsigned char sleepMode)
This function switches the microcontroller in the power consumption mode according to the datasheet of the ATmega16. As parameter enter the desired contents of the MCUCR Register.

void setExternalInterrupt(unsigned char status)
This function enables the external interrupt 0, it is connected to the joystick and push button. A 1 as parameter sets the 6th bit of the GICR, a 0 clears it. For details on the GICR, please see the section "External Interrupts" of the datasheet ATmega16.

void setRegister(unsigned char regMemoryAddress, unsigned char value)
This function writes the unsigned char value to the memory address of a certain register. For example 0x38 is the memory address of PORTB, setRegister(0x38,0x01) writes the value 0x01 to PORTB. With twiSetRegister (unsigned char regMemAddr, unsigned char value) it is possible to manipulate registers of the Flamingo's MCU from the STK500 over the TWI bus, provided that *flamingo.c* is running on the Flamingo or its MCU knows how to interpret the incoming data.

3.4.5 A/D Conversion

unsigned int getAdcValue(unsigned char adcInput)
This function returns the result of the A/D conversion configured by the byte adcInput. The byte adcInput can be set according to the datasheet of the ATmega16, Table 84 "Inputchannel and Gain Selection". Possible ADMUX settings only use channel 0 and channel 1. For example request A/D conversion on channel 0 (PINA.0) set adcInput = 0, channel 1 (PINA.1) set adcInput = 1.

Stk_ui.h
The library STK_UI.h contains functions that provide access to the Flamingo's user interface devices over the TWI bus. Through these functions it is possible to manipulate the user interface from the STK500's/battery-board microcontroller.

3.4.6 LED

void twiOutputLed(unsigned char illumination)
Switching on/off the green and red LED of the Flamingo over the TWI interface from the STK500 – the LSB of the parameter is set to 1, the red LED is switched on, when the second LSB of the parameter is set to 1 the green LED is switched on.

3.4.7 Printing to LCD

void twiOutputLcdChar(unsigned char x-position, unsigned char y-position, unsigned char myCharacter)
This function will cause the Flamingo to print a character at the given position to the LCD.

void twiOutputLcdCursor(unsigned char x-position, unsigned char y-position, unsigned char cursor)
This function will cause the Flamingo's microcontroller to change the position and appearance of the cursor on its LCD via the TWI interface. The parameters are the normal unencoded position values.

*void twiOutputLcdString(unsigned char x-position, unsigned char y-position, unsigned char *stringPointer)*
This function causes the remote microcontroller to print a string of characters at the position (x,y).

void twiOutputSound(unsigned int freqInHz, unsigned int durationInMilliSec)
This function causes the remote microcontroller to make a sound at the given frequency and duration.

void twiOutputLcdDecimalInt(unsigned char x-position, unsigned char y-position, int intValue)
This function causes the Flamingo's microcontroller to print the integer *intValue* to the LCD.

void twiClearLcd()
Sends the command to clear the LCD to the Flamingo’s microcontroller.

3.4.8 Sleepmodes and External Interrupt Control

void twiSetSleepMode(unsigned char sleepMode)
This function sends the desired content of the MCUCR Register to the remote microcontroller.

void twiSetExternalInterrupt(unsigned char status)
This function manipulates the 6th bit of GICR. For a “1” as parameter the bit will be set, for a “0” the bit will be cleared. If the bit is cleared the remote microcontroller will not respond to joystick or push button events because its external interrupt 0 is disabled.

void twiSetRegister(unsigned char regMemAddr, unsigned char value)
This function causes the Flamingo's microcontroller to write the unsigned char value to the given memory address. For example: to make the remote MCU write a 0xFF to its PORTB use *twiSetRegister(0x38,0xFF)*, however the memory address of PORTB differs from its special function-register's number 0x18. You can find the memory addresses of all registers in the register summary in the datasheet ATmega16.

3.4.9 A/D Conversion

unsigned int twiGetAdcValue(unsigned char adcInput)
This function sends the request for an A/D conversion result according to the MUX configuration in adcInput to the remote microcontroller, usually the one on the Flamingo.

3.4.10 General TWI Functions

void twiInit(unsigned char mySlaveAddress, unsigned char myBitRateDenominator)
This function activates the TWI engine by writing 0x45 to TWCR. It also sets the microcontrollers own slave address in TWAR and writes the bit rate denominator into the register TWBR.

void setRemoteAddress(unsigned char addr)
This will set the address of the remote MCU where all data will be sent to.

void sendBytes(unsigned char arrayLength)
This function sends data to the remote microcontroller. The parameter indicates the number of the bytes that will be sent. Please use the function loadSendBuffer to store the information in the array *sendBuffer*.

*void loadSendBuffer(unsigned char arrayLength, unsigned char *ptr)*
This function stores arrayLength bytes from an array represented by the pointer *ptr into the array sendBuffer from where the routine sendBytes will send them.

void setTwiState(eTwiStateType state)
This function can be used to set the transmission mode such as Master transmitter or Master receiver. It is important not to start a new TWI action until the interface finishes the last one. The enumeration eTwiState contains the mode IDLE which is the non-active mode. With a while-loop that is waiting for the TWI hardware to become idle it is possible to let the last bus transaction be completed before starting a new one.

eTwiStateType getTwiState()
This function returns the current state of the TWI communication. The variable *twiState* is changed to a different value than *IDLE*, when the MCU is sending or receiving data. This mechanism prevents collisions and arbitration problems on the bus.

unsigned char newBytesReceived()
This function returns the number of bytes that have been received during the last transaction on the TWI bus. When this function is called after some bytes have been received it sets the variable *bytesReceived* again to zero. If the microcontroller expects data from the remote one, it can be put into a while-loop that is observing the return value of this function.

*unsigned char *getReceiveArrayPointer()*
This function returns a pointer directing to the array *receiveBuffer*. In this array currently received data is stored. Through this function it is possible to get access to the received data.

3.4.11 Interrupt Service Routines

interrupt [18] void twi_isr(void)
This interrupt routine handles the communication between the two microcontrollers over the TWI bus. It is called when the TWI-status register TWSR is updated.

interrupt [2] void ext_int0_isr(void)
This interrupt function is called if the PIND.2 is pulled to GND. This happens when the joystick is moved or the push button is pressed.

Section 4

Schematics of Flamingo™ Board

Figure 4-1. Expand 0 Connector

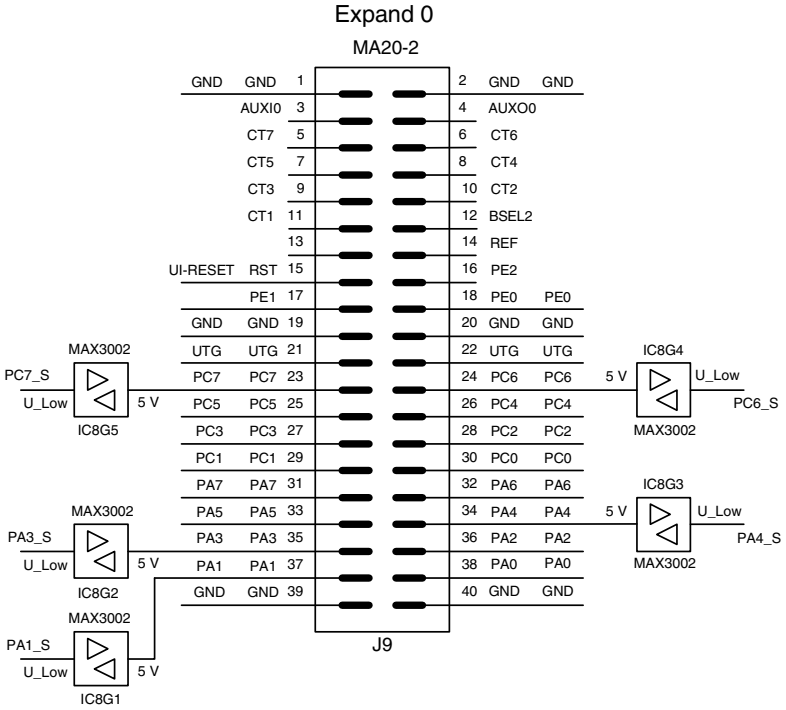


Figure 4-2. Expand 1 Connector

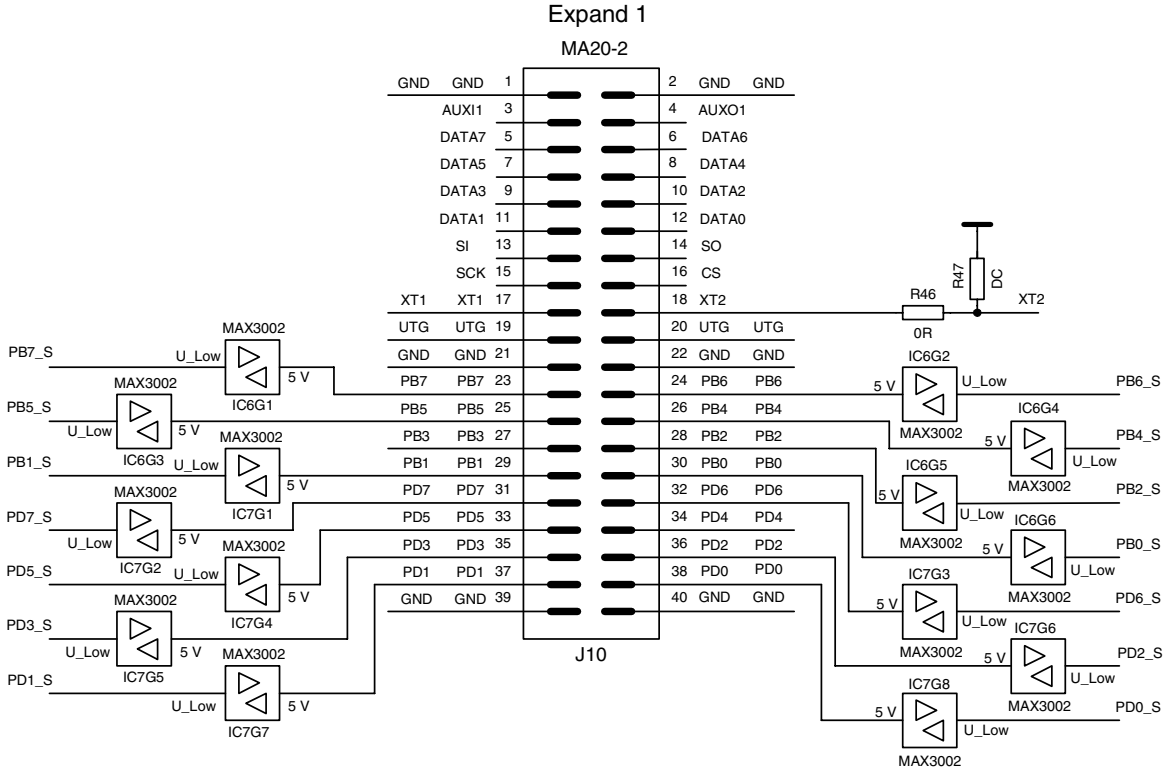


Figure 4-3. Power Supply Section

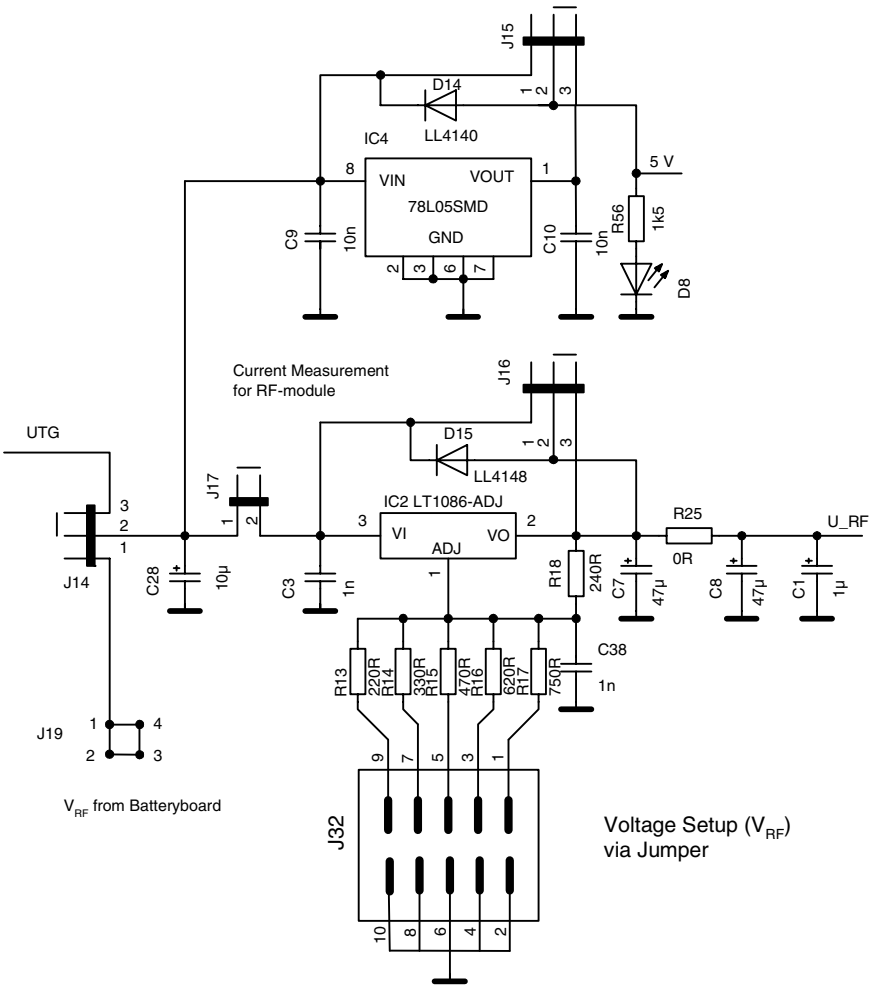


Figure 4-4. Level Shifter Supply

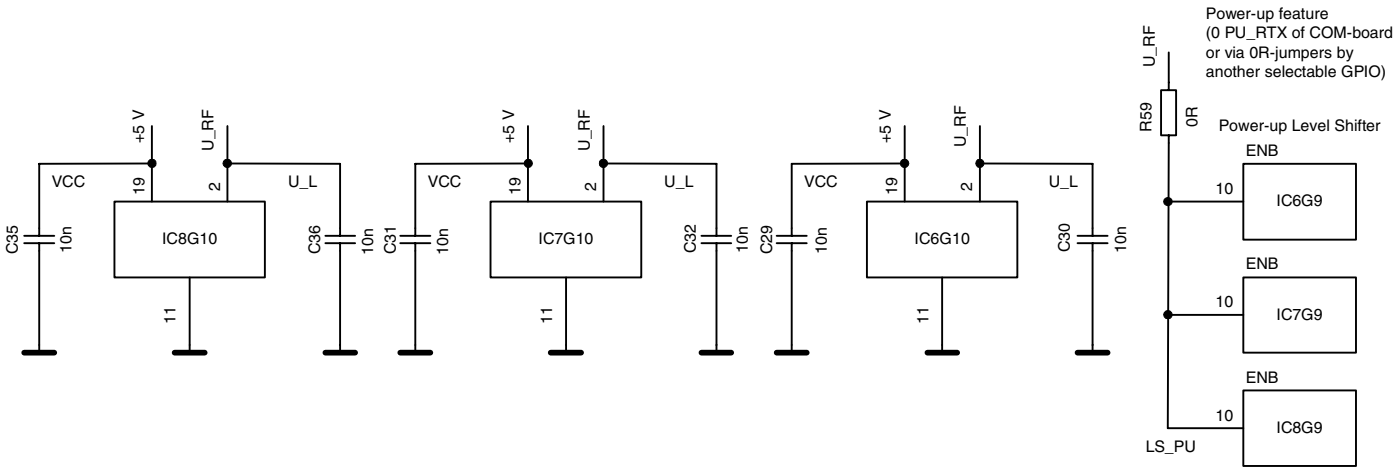


Figure 4-5. User Interface Circuit

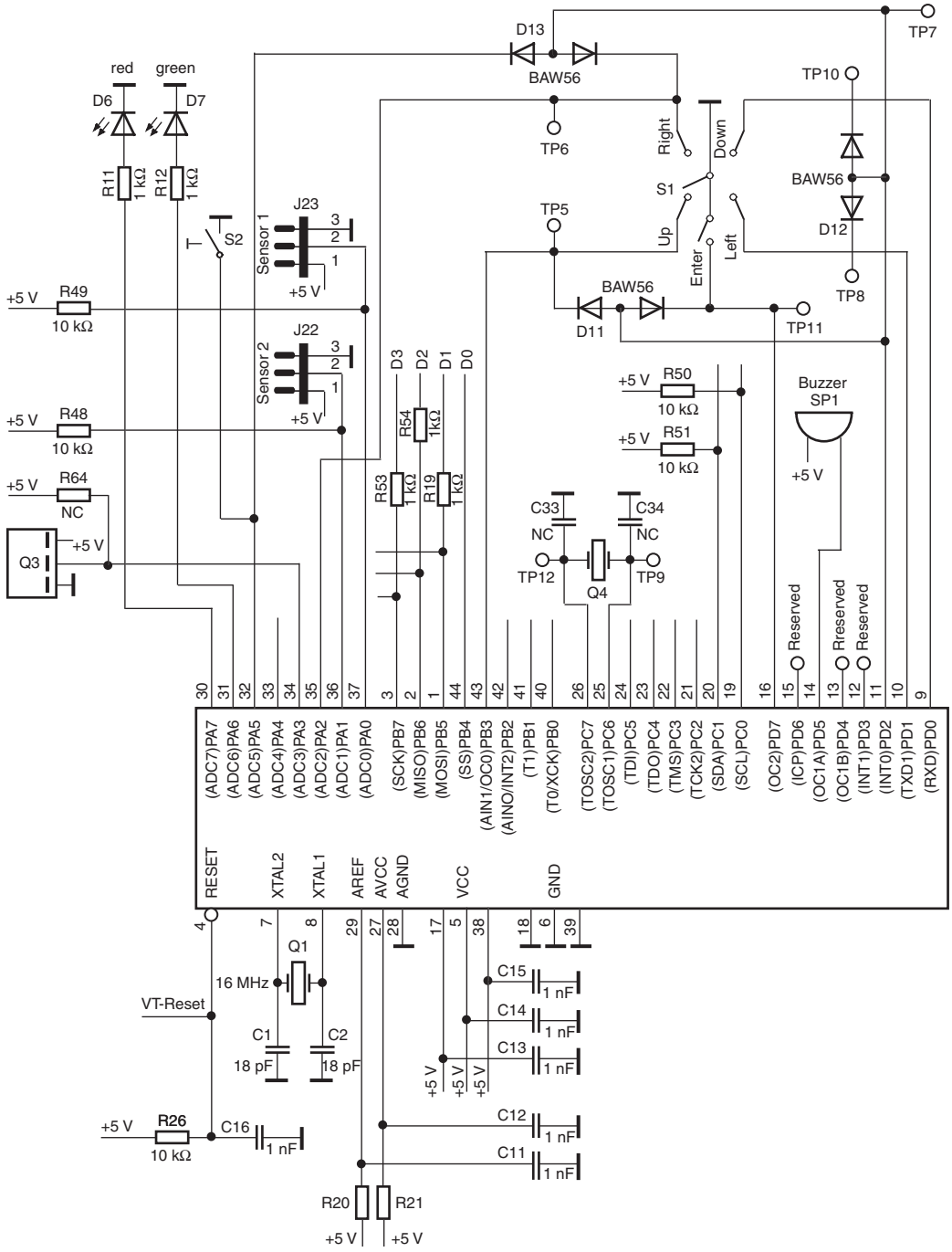


Figure 4-6. User Interface Display Circuitry

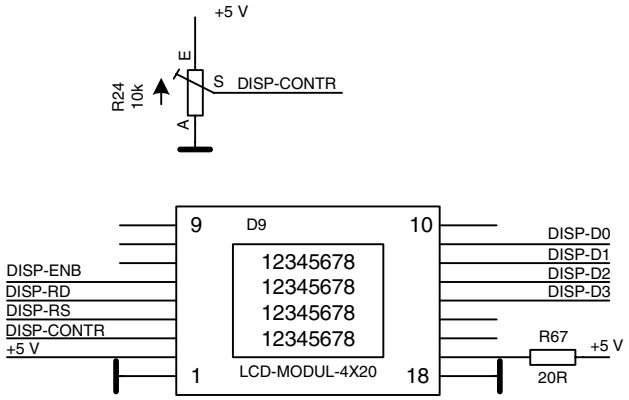
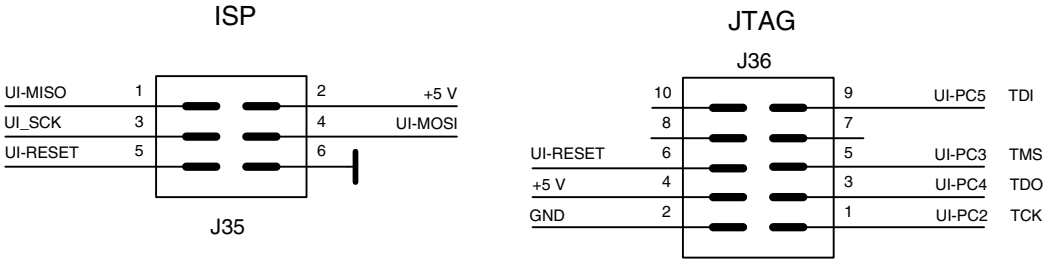


Figure 4-7. JTAG + ISP Pinout



Section 5

Connectors for RF Modules

Figure 5-1. Connectors for ATA575x, ATA574x, ATA588x RF Chipsets

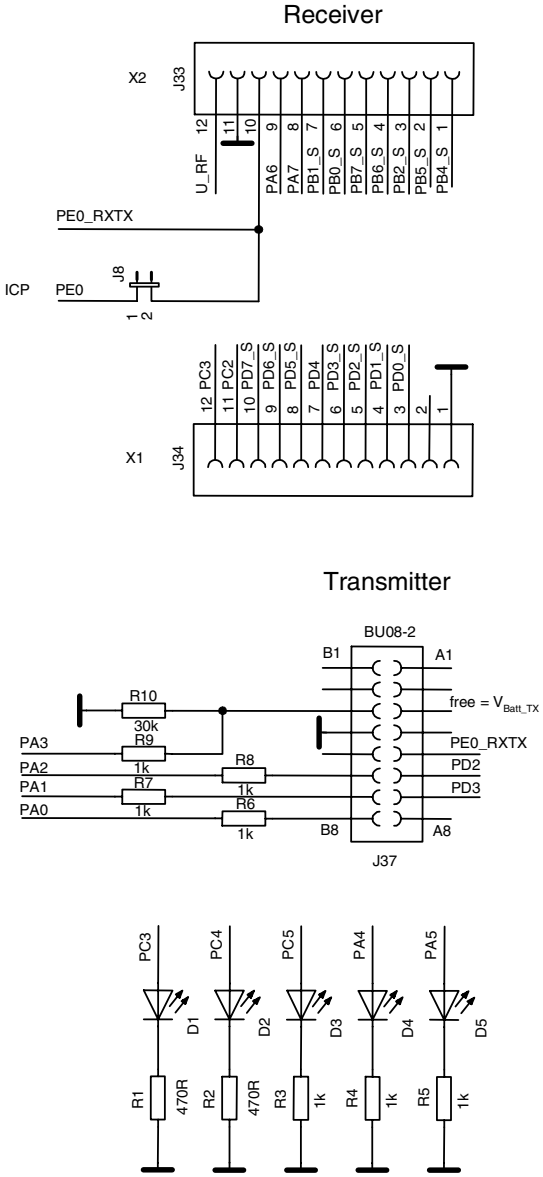


Figure 5-2. Connectors for AT86R211, AT86RF211s RF Chipsets

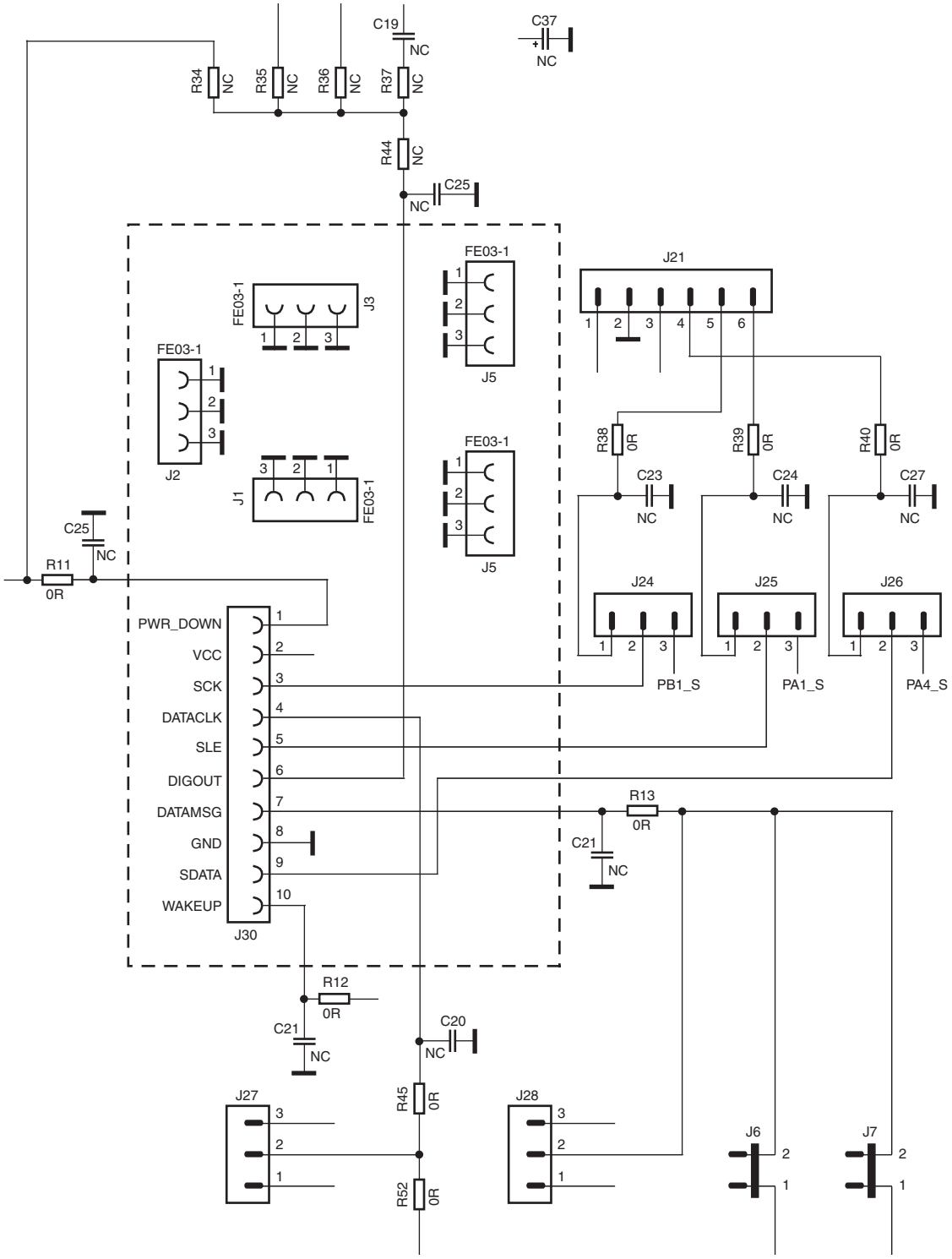


Figure 5-3. Connectors for T2080x, ATR2806, ATR2406 RF Chipsets

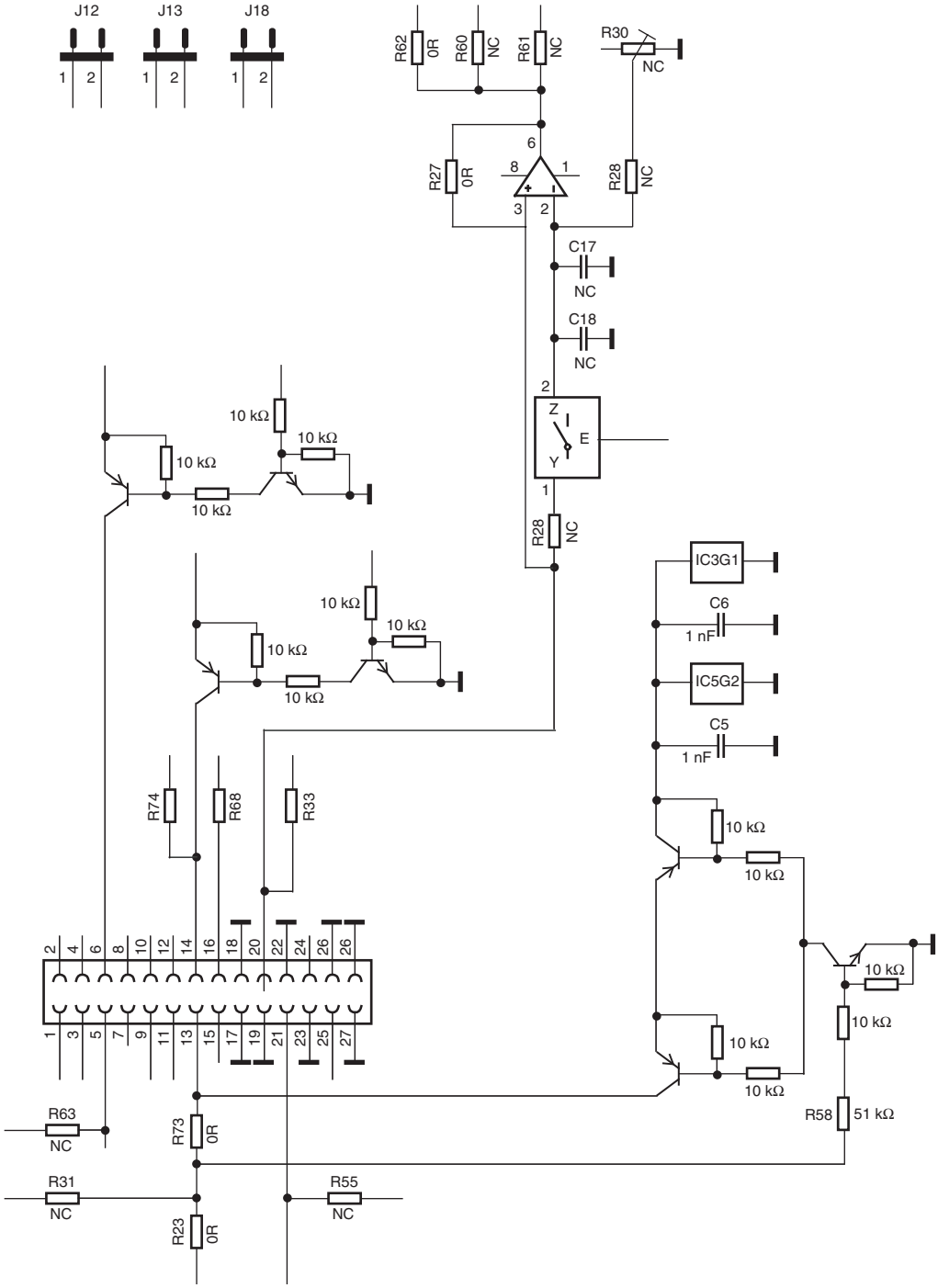


Figure 5-4. Additional Level Shifters

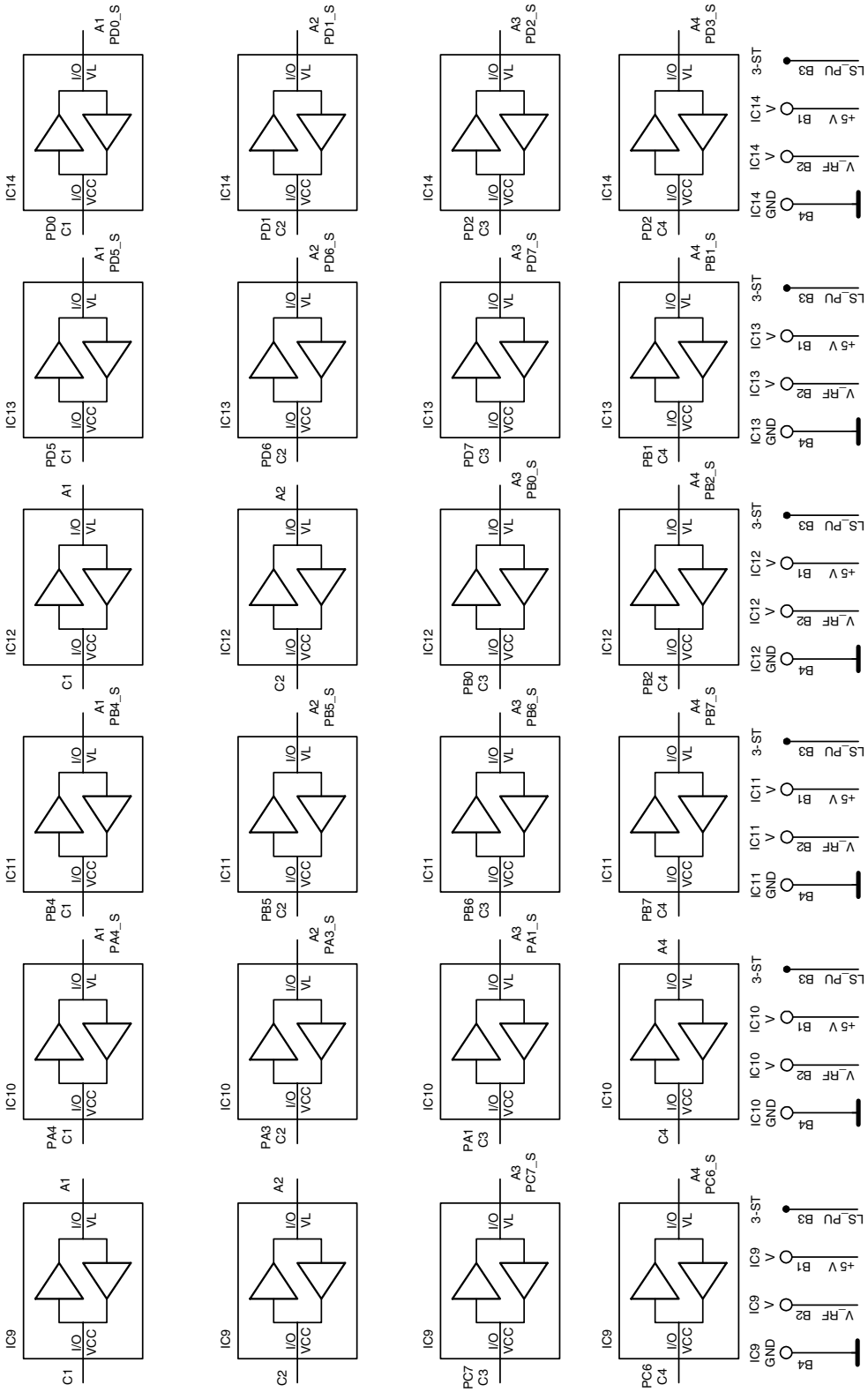


Figure 5-5. Expand 0 Connector

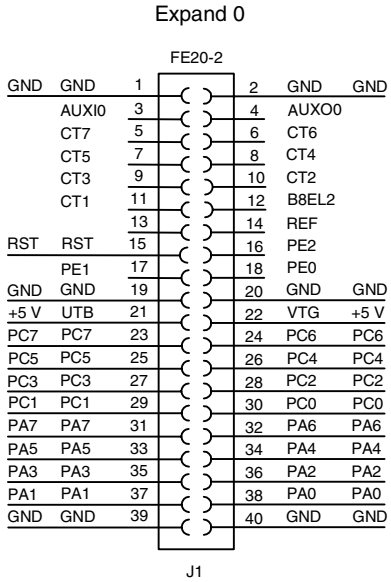


Figure 5-6. Expand 1 Connector

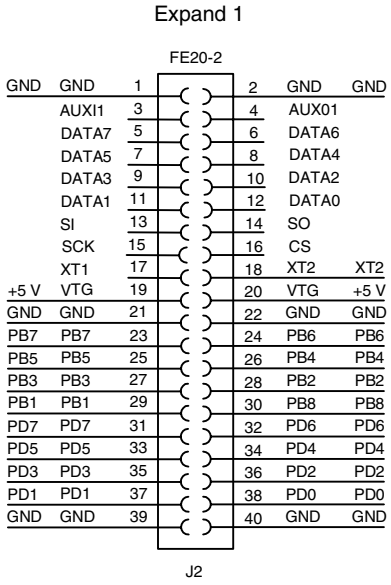


Figure 5-7. Pinout for JTAG + ISP Connector

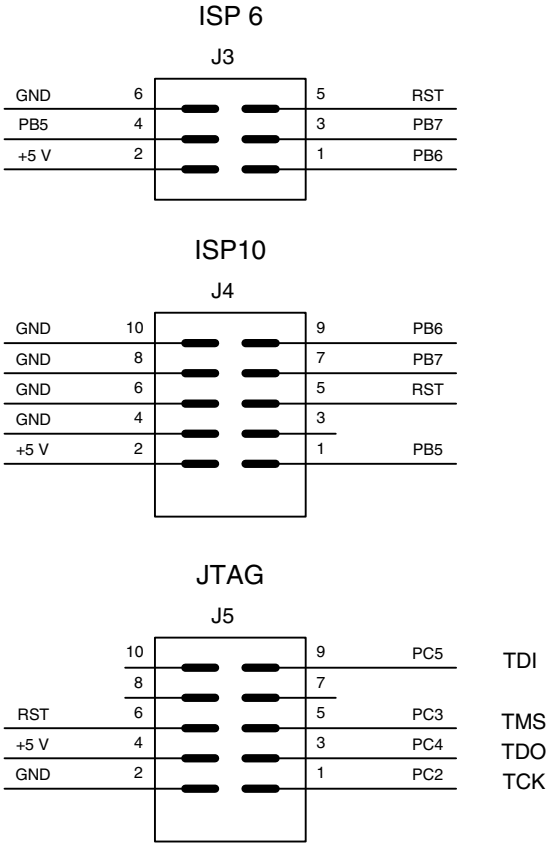


Figure 5-8. Optional Step-up DC-supply

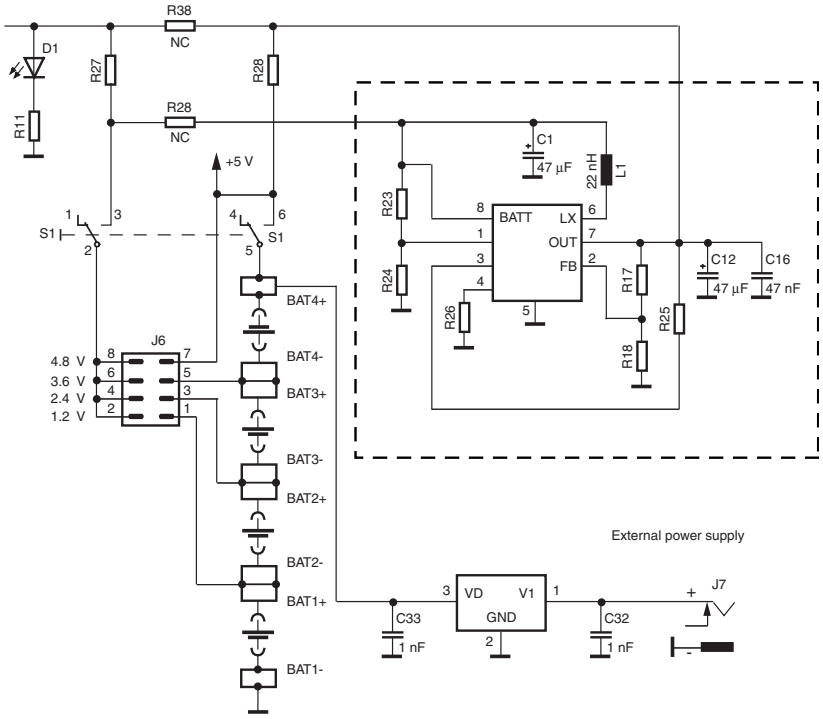


Figure 5-9. ATmega16 Reset, Analog Reference, Supply Circuitry

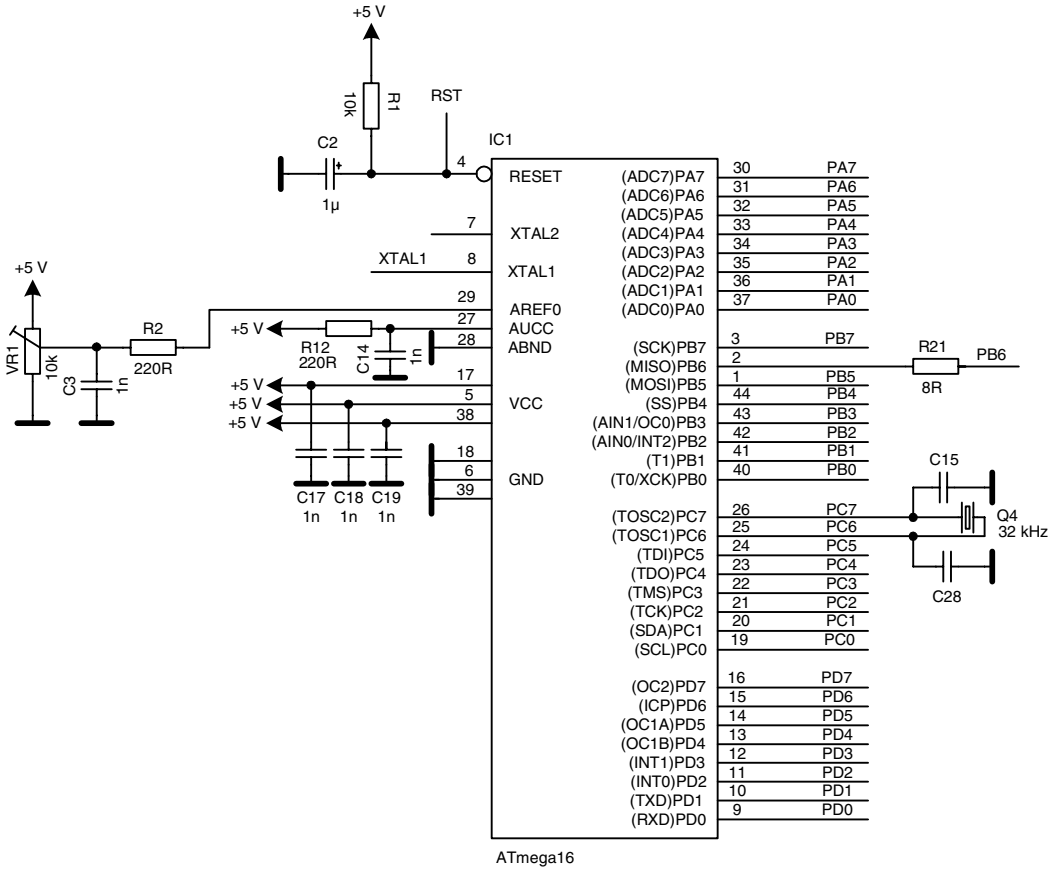


Figure 5-10. Optional Serial Interface Circuit

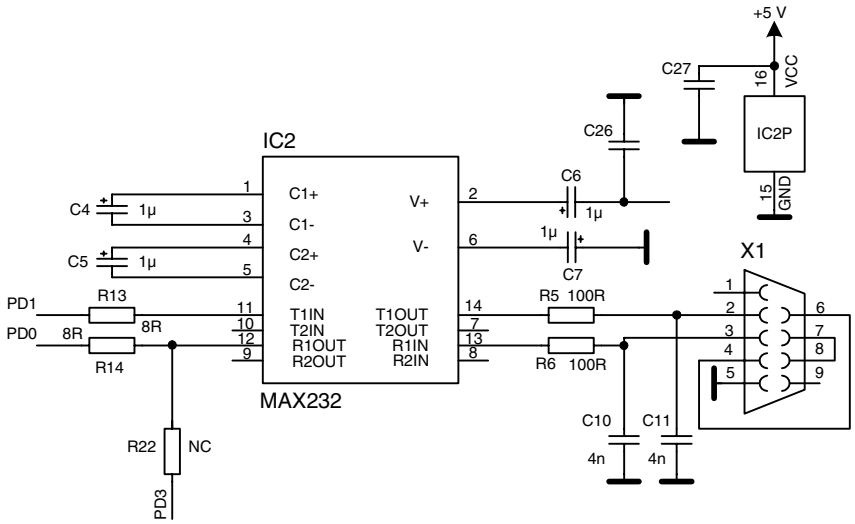
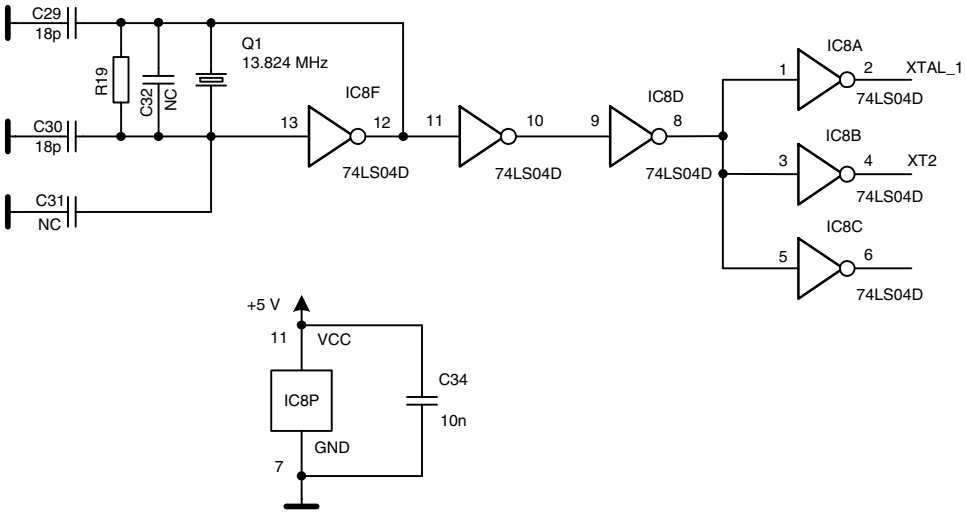


Figure 5-11. Reference Clock Oscillator Circuit



Section 6

Abbreviations

MCU	MicroController Unit
TWI	Two-Wire-Interface
UID	User-Interface-Device

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© Atmel Corporation 2005. All rights reserved. Atmel®, logo and combinations thereof, and others, are registered trademarks, and Everywhere You Are℠ and others are the trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Printed on recycled paper.

4825C–ISM–02/05 /xM