

# RS232-ADC16/24

## Manual

**Version 1.13**  
Copyright taskit GmbH 2009  
[www.taskit.de](http://www.taskit.de)

# Table of contents

1 Features.....	3
2 Introduction.....	3
3 Bringing into service.....	4
4 Application Sample.....	5
5 Frame layout.....	6
5.1 LRC calculation.....	7
6 Commands.....	8
6.1 Read Holding Registers.....	8
6.2 Read Input Registers.....	8
6.3 Write Single Register.....	9
6.4 Write Multiple Registers.....	9
6.5 Calibrate device.....	10
6.6 Save calibration data.....	10
6.7 Error Response.....	11
7 Register description.....	12
7.1 Holding registers.....	12
7.2 Input registers.....	16
8 Technical Details.....	17
8.1 RS232 connector.....	17
8.2 DSUB-25 connector.....	17
8.3 Electrical characteristics.....	18
8.4 Reaction time.....	19
8.5 Power consumption.....	20
8.6 Baud rate error.....	21
8.7 ADC.....	21

## 1 Features

- C8051F350/C8051F352 microcontroller at 3 – 49 Mhz
- RS232 interface
- Eight analog 16/24-bit inputs
- Eight digital 3.3V IOs (5V tolerant)
- Supply voltage range from 4.5V – 15V (absolut max. 20V)
- Provides 3.3V power supply for sensors
- Seperated analog ground
- Extended temperature range (-40°C – 85°C)
- Small instruction set
- Recalibration possible
- Access via C-library

## 2 Introduction

RS232-ADC16/24 device is designed to give simple access to external sensors and peripherals over a standard RS232 connection. There is no need for any additional drivers when using Windows, DOS or Linux systems. Microcontroller only need a simple implementation of their UART interface.

Instructions send to the device have an human readable form allowing manual traffic checking for debugging proposes. To integrate RS232-ADC16/24 into your application, a C-library is provided with the device bearing the work of protocol parsing and packet construction.

The device comes in two versions, one with 16-bit and one with 24-bit ADC channels. These versions are named respectively to their ADC channel resolution RS232-ADC16 and RS232-ADC24.

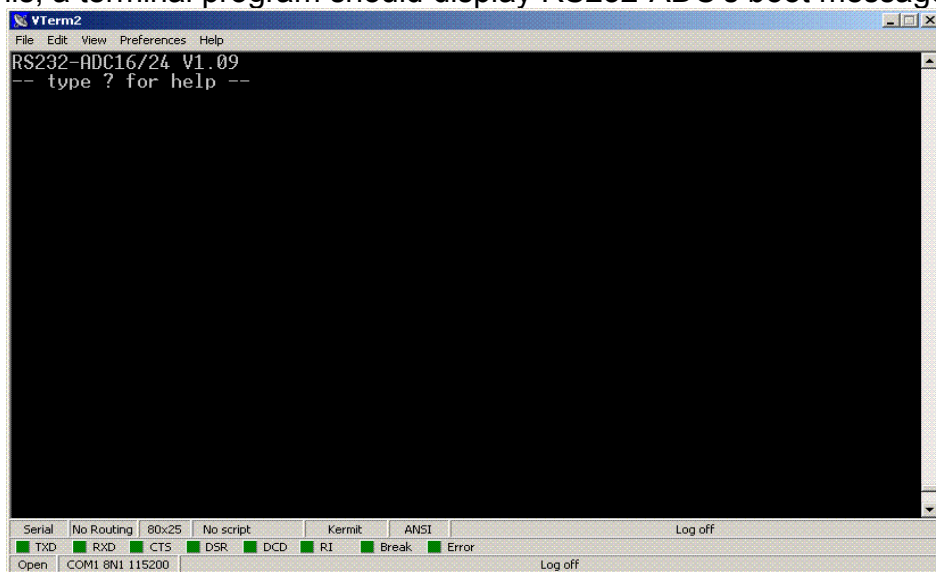
This document contains information about hardware issues, the software protocol and resulting register mappings. It provides extra information for new firmware versions 1.11 & 1.12.

### 3 Bringing into service

The following steps illustrate how to setup the device correctly.

1. Connect DSUB-9 adaptor to serial port.
2. Configure serial port to 115200 baud, 8N1 (8 bit data, no parity, 1 stop bit).
3. Power the device.  
Use VIN (pin 24) and GND (pin 18) pins of DSUB-25 connector. (15V max)

After this, a terminal program should display RS232-ADC's boot message.

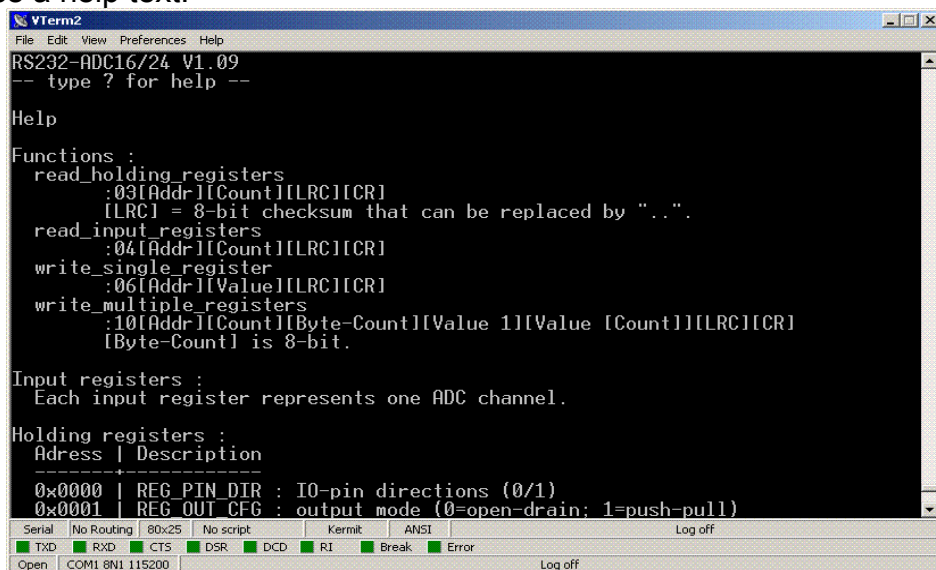


```

VTerm2
File Edit View Preferences Help
RS232-ADC16/24 V1.09
-- type ? for help --
Serial No Routing 80x25 No script Kermit ANSI Log off
TXD RXD CTS DSR DCD RI Break Error
Open COM1 8N1 115200 Log off

```

Referring to the advice given in the boot message sending a questionmark will produce a help text.



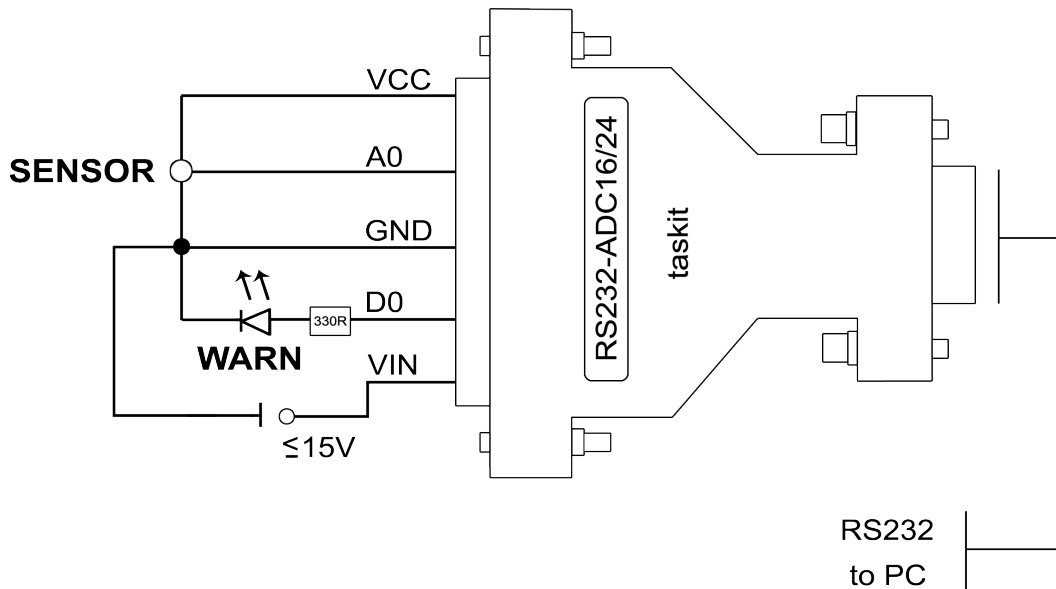
```

VTerm2
File Edit View Preferences Help
RS232-ADC16/24 V1.09
-- type ? for help --
Help
Functions :
read_holding_registers
:03[Addr][Count][LRC][CR]
[LRC] = 8-bit checksum that can be replaced by "...".
read_input_registers
:04[Addr][Count][LRC][CR]
write_single_register
:06[Addr][Value][LRC][CR]
write_multiple_registers
:10[Addr][Count][Byte-Count][Value 1][Value [Count]][LRC][CR]
[Byte-Count] is 8-bit.
Input registers :
Each input register represents one ADC channel.
Holding registers :
Address | Description
-----+-----
0x0000 | REG PIN DIR : IO-pin directions (0/1)
0x0001 | REG OUT_CFG : output mode (0=open-drain; 1=push-pull)
Serial No Routing 80x25 No script Kermit ANSI Log off
TXD RXD CTS DSR DCD RI Break Error
Open COM1 8N1 115200 Log off

```

## 4 Application Sample

To illustrate the use of RS232-ADC16/24 – in special regard to VCC and VIN – you might consider this schematic overview.



The task is to obtain sensor's data and transfer it to a monitoring software. In case of critical or unusal conditions a nearby LED should indicate this. Obviously, up to eight pairs of sensor and LED can be controlled using one RS232-ADC16/24.

As described earlier, the device is powered via VIN at a maximum of 15V. The device's internal voltage regulator provides 3.3V at VCC to power external sensors. According to this, VCC should **never** be used to power RS232-ADC16/24. Furthermore the range of the analog input (0 – 2.5V) has to be taken in account.

## 5 Frame layout

This chapter specifies the used frame format. A frame does not contain more than one command; each command is send together with all of its parameters.

Parameters larger than eight bits are expected in big-endian format, which means the most significant byte (MSB) is send before the least significant byte (LSB).

On serial line each byte is send with eight bits, one start bit and one stop bit. There are no parity bits involved in communication. The standard baud rate is 115200 baud. At the moment no handshaking is used because the host is intended to wait for a response before sending a new request. Handshake can be implemented in further versions as RS232-ADC16/24 provides a fully populated RS232 connector.

A frame is encoded in an human readable ASCII format. Therefore each byte of the frame's data is converted into two HEX digits. The converted frame then consists of one start character, the command and parameters in HEX digits, a packet LRC – also in HEX digits – and an end character.

Byte <sub>1</sub>	Byte <sub>2</sub>	Byte <sub>3</sub>	Byte <sub>4</sub>	Byte <sub>5</sub>	...	Byte <sub>n-2</sub>	Byte <sub>n-1</sub>	Byte <sub>n</sub>	
:	Command		Parameter <sub>1</sub>				LRC		<CR>

The end character is a "carriage return" (<CR>, '\r') resulting in hex code 0x0D. Any additional "new line" (<NL>, '\n', 0x0A) is ignored by device and added to any response.

**Example** : Read two ADC channels starting at 0x0001

Request :

```
:0400010002..<CR><NL>
```

Response :

```
:0404XXXXYYYY..<CR><NL>
```

**Note** : The first parameter is the byte count of read data, XXXX the value of channel1, YYYY the value of channel2. LRC checking in requests can be avoided by setting LRC to "..", replies always have a LRC although it is set to ".." in this example.

## 5.1 LRC calculation

The packet LRC is the two's complement of the sum over the packet's parameter and command bytes.

$LRC = -(\sum data_n + command)$ ; with  $data_n$  as **binary** byte values of the packet's parameter list and command as **binary** value.

The correct LRC for the sample request above would have been 0xF8.

In regard to this including the LRC of a received packet itself into calculation will result in a sum equal to 0, otherwise a transmission error has occurred.

**Note** : Each binary data byte is transmitted as two HEX digits which have to be transformed into binary form **before** adding them.

## 6 Commands

The device uses a MODBUS similar command set, assuming all parameters as 16-bit values.

The following commands are provided.

### 6.1 Read Holding Registers

A contiguous block of 16-bit software holding registers can be read at once. Addressing starts at 0, count is greater than 0 and less or equal 125.

Command : Read Holding Registers		
Function code	1 byte	0x03
Start address	2 bytes	0x0000 to 0xFFFF
Register count	2 bytes	0x0000 to 0x007D

If the function succeeds the response contains the read registers.

Response : Read Holding Registers		
Function code	1 byte	0x03
Byte count	1 byte	(Register count) * 2
Register data	(Register count) * 2 bytes	

Otherwise an error response is send.

### 6.2 Read Input Registers

This function is the equivalent to Read Holding Registers corresponding to input registers.

Command : Read Input Registers		
Function code	1 byte	0x04
Start address	2 bytes	0x0000 to 0xFFFF
Register count	2 bytes	0x0000 to 0x007D

If the function succeeds the response contains the read registers.



Response : Read Input Registers		
Function code	1 byte	0x04
Byte count	1 byte	(Register count) * 2
Register values	(Register count) * 2 bytes	

Otherwise an error response is send.

### 6.3 Write Single Register

Single holding registers can be written using this function. The function code is followed by a 16-bit register address and a 16-bit register value.

Command/Response : Write Single Register		
Function code	1 byte	0x06
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	

If the function succeeds the response is equal to the received command otherwise an error response is send.

### 6.4 Write Multiple Registers

If more than one holding registers have to be written, this function can be used. A block of register values follows function code, address and register count. Up to 123 registers can be written simultaneous.

Command : Write Multiple Registers		
Function code	1 byte	0x10
Register address	2 bytes	0x0000 to 0xFFFF
Register count	2 bytes	0x0001 to 0x007B
Byte count	1 byte	(Register count) * 2
Register values	(Register count) * 2 bytes	

If the function succeeds the corresponding response is send.

Response : Write Multiple Registers		
Function code	1 byte	0x10
Register address	2 bytes	
Register count	2 bytes	

Otherwise an error response is send.

## 6.5 Calibrate device

This command calibrates the device for external circuits. Analog channels 0 and 1 have to be set to the following levels during calibration.

Channel 0 must be connected to analog ground.

Channel 1 must be connected to the highest expected valid level.

**Note** : The device is already calibrated at delivery.

Command/Response : Calibrate Device		
Function code	1 byte	0x64

If the function succeeds the response is equal to the received command otherwise an error response is send.

## 6.6 Save calibration data

Calibration data is loaded at system reset. If another calibration data is needed by default, use this command after device calibration (see above). A key is needed to unlock this operation.

Since firmware version 1.09 this function also saves the baud rate setting to provide to boot with a customized baud rate.

**Note** : Saving calibration data will overwrite factory settings! Do not use this function after having called „Calibrate device“ accidentally.

Command/Response : Save Calibration Data		
Function code	1 byte	0x65
Key	2 bytes	0xABCD

If the function succeeds the response is equal to the received command otherwise an error response is send.

## 6.7 Error Response

Every function that leads into an error condition responds with a two-byte answer containing the indicated function code and an additional error code. Indication is achieved by setting the function code's highest bit.

Error Response		
Indicator	1 byte	0x80   (function code)
Error code	1 byte	

Error code	Description
1	Illegal function call
2	Address out of range
3	Inconsistent data

## 7 Register description

The device's configuration is handled via read/writeable holding registers. PIO data is also available at these holding registers. ADC channels appear as read-only input registers.

### 7.1 Holding registers

Register map : holding registers		
Address	Name	Description
0x0000	REG_PIN_DIR	Configuration of pin input/output direction
0x0001	REG_OUT_CFG	Output mode selection
0x0002	REG_OUT_VAL	Output pin levels
0x0003	REG_IN_VAL	Input pin levels
0x0004	REG_VERSION	Firmware version <sup>1</sup>
0x000D	REG_ADC_DEC	ADC precision
0x000E	REG_BAUD	Baud rate selection
0x000F	REG_SYSCLK	System clock selection

<sup>1</sup> implemented in version 1.11.

#### REG\_PIN\_DIR 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	Dir <sub>7</sub>	Dir <sub>6</sub>	Dir <sub>5</sub>	Dir <sub>4</sub>	Dir <sub>3</sub>	Dir <sub>2</sub>	Dir <sub>1</sub>	Dir <sub>0</sub>

Dir<sub>n</sub> selects between pin input and pin output. A set bit configures the corresponding pin D<sub>n</sub> as output. Input pins have are internally pulled to high level. The default value of this register is 0x0000.

#### REG\_OUT\_CFG 0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	OM <sub>7</sub>	OM <sub>6</sub>	OM <sub>5</sub>	OM <sub>4</sub>	OM <sub>3</sub>	OM <sub>2</sub>	OM <sub>1</sub>	OM <sub>0</sub>

OM<sub>n</sub> programs its corresponding pin D<sub>n</sub> into either push-pull or open-drain mode if D<sub>n</sub> is configured as output pin. A logic zero selects open-drain mode. Any changes to input pins are ignored. The default value of this register is 0x0000.

**REG\_OUT\_VAL 0x0002**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	Out <sub>7</sub>	Out <sub>6</sub>	Out <sub>5</sub>	Out <sub>4</sub>	Out <sub>3</sub>	Out <sub>2</sub>	Out <sub>1</sub>	Out <sub>0</sub>

Out<sub>n</sub> sets the pin level of the corresponding pin D<sub>n</sub>. Any changes to input pins are ignored. The default value of this register is 0x00FF.

**REG\_IN\_VAL 0x0003**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	In <sub>7</sub>	In <sub>6</sub>	In <sub>5</sub>	In <sub>4</sub>	In <sub>3</sub>	In <sub>2</sub>	In <sub>1</sub>	In <sub>0</sub>

Input levels at pin D<sub>n</sub> appear in bit In<sub>n</sub> if D<sub>n</sub> is programmed as input. Otherwise the actual output level of D<sub>n</sub> can be read back. Any writes to this register are ignored.

**REG\_VERSION 0x0004**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Major version								Minor version							

Since firmware version 1.11 this register is readonly and holds the device's version number. Earlier versions contain 0 as long as this register is not overwritten.

**REG\_ADC\_DEC 0x000D**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC cycles															

Selects the number of clock<sub>ADC</sub> cycles used to measure the level of an ADC input. Increasing this value will result in more accurate measurement values, but will decrease the maximal output frequency of the ADC. Accepted values are between 5 and 15.<sup>1</sup> Any register setting beyond these limits will result in 11, which is the default and initial setting of this register.<sup>2</sup>

Output frequency is calculated the following way :

$$f_{\text{output}} = \text{clock}_{\text{SYS}} / 3 / 128 / 2^{\text{ADC cycles}} \cdot 3$$

**Note** : Output frequency is also limited by the selected baud rate. When using  $\text{clock}_{\text{SYS}}$  at 49 MHz the clock used by ADC is 24.5 MHz.

<sup>1</sup> Firmware version 1.12 extends range of accepted values to 5 – 15, for earlier versions this range is 5 – 11.

<sup>2</sup> For devices with at least firmware version 1.11 – default value for earlier versions is 5.

<sup>3</sup> Firmware versions below 1.11 use this formula :

$$f_{\text{output}} = \text{clock}_{\text{ADC}} / 3 / 2^{\text{ADC cycles}}, \text{ with } \text{clock}_{\text{ADC}} = 19200 \text{ Hz.}$$

### REG\_BAUD 0x000E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Baud rate select															

Selects the baud rate used for serial transmission. The baud rate can be chosen in five steps between 9600 and 115200 baud. Default baud rate is 115200. Any register setting above 4 will be interpreted as 4. After successful command execution the boot message will be send using the new baud rate. However the reply to the command is send with the old settings.

REG_BAUD	Resulting baud rate
0	9600
1	19200
2	38400
3	57600
4	115200

**REG\_SYSCLK 0x000F**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
clock <sub>SYS</sub> select															

The system clock can be chosen in five steps from 3.0625 MHz to 49 MHz. Any register setting outside range will result in default clock<sub>SYS</sub> at 12.25 MHz. Altering the system clock will cause a short delay before sending the boot message. All pin and register settings remain untouched by clock change. Changing clock and baud rate together (by Write Multiple Registers) will result in sending one boot message not two. It is suggested wait for device's reboot by trying to read any holding register.

Using 12.25 MHz as system clock is recommended since this offers best results in dependence to ADC output rate. Increasing clock<sub>SYS</sub> will provide better reaction time and higher output rates at cost to accuracy. Decreasing clock<sub>SYS</sub> mainly lowers power consumption.

REG_SYSCLK	Resulting system clock in MHz
0	3.0625
1	6.125
2	12.25
3	24.5
4	49

**Note** : Setting REG\_SYSCLK to 0 will reduce the maximal baud rate to 57600.

## 7.2 Input registers

Register map : input registers		
Address	Name	Description
0x0000	REG_ADC0_VAL	16-bit value measured at A <sub>0</sub>
0x0001	REG_ADC1_VAL	16-bit value measured at A <sub>1</sub>
0x0002	REG_ADC2_VAL	16-bit value measured at A <sub>2</sub>
0x0003	REG_ADC3_VAL	16-bit value measured at A <sub>3</sub>
0x0004	REG_ADC4_VAL	16-bit value measured at A <sub>4</sub>
0x0005	REG_ADC5_VAL	16-bit value measured at A <sub>5</sub>
0x0006	REG_ADC6_VAL	16-bit value measured at A <sub>6</sub>
0x0007	REG_ADC7_VAL	16-bit value measured at A <sub>7</sub>

A measurement at A<sub>n</sub> is started each time "Read Input Registers" is called with address n. If multiple registers are read each affected ADC channel is treated sequentially. The function blocks until all measures are finished before the result is send back.

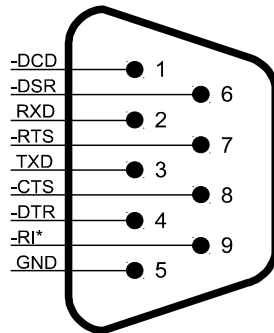
Access to the lower eight bits of an 24-bit conversion is possible by reading input registers 0x0008 to 0x000F (named REG\_ADC<sub>0</sub>\_LOW to REG\_ADC<sub>7</sub>\_LOW), each containing the low byte of the 24-bit value last measured at A<sub>0</sub> to A<sub>7</sub>. For RS232-ADC16 modules these registers are always read 0.

$$\text{adc\_value}_{24} = (\text{REG\_ADC}_x\_\text{VAL} \text{ shl } 8) + \text{REG\_ADC}_x\_\text{LOW}$$



## 8 Technical Details

### 8.1 RS232 connector

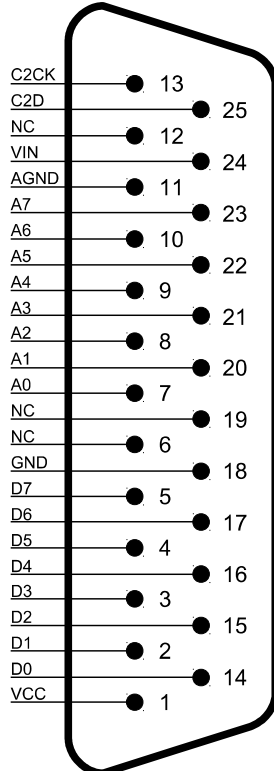


Connection between host and RS232-ADC16/24 device is realized with a female 9-pin D-SUB connector. The pin layout is consistent with the standard RS232 pin mapping except -RI\* which can either be used to detect the device or as device's alternative power supply pin.

Therefore -RI\* is internally wired to VIN. As result, if -RI\* pin is connected to the host, input voltage of device's power supply is limited to 15V maximum due to the RS232 specification.

**Note** : Handshake currently is not used.

### 8.2 DSUB-25 connector



Analog sensors and digital peripherals are to be wired at the male 25-pin D-SUB connector. In addition to the eight digital IOs D0 to D7 and the eight analog inputs A0 to A7 this port provides a 3.3V power supply for connected sensors and peripherals, so the RS232-ADC16/24 device's voltage regulator can be used to power these external components eliminating the need of additional voltage regulators.

Pins 6, 12 and 19 currently are not used but reserved for possible extensions and should not be connected.

Lines C2D and C2CK allow programming of the used C8051F35x microcontroller.

AGND provides a separated analog circuit.

**Note** : Power supply over VIN must not exceed 15V if -RI\* pin of RS232 connector is connected to a RS232 counterpart.

### 8.3 Electrical characteristics

Digital IO		
	Min	Max
Output high level	2.5 V	3.3 V
Output low level	-	1.0 V
Input high level	2.0 V	5.0 V
Input low level	-	0.8 V
Output current <sup>1</sup>	-	75 mA
Analog Input		
	Min	Max
Input voltage <sup>2</sup>	0 V	2.5 V (analog max.) 3.6 V (elec. max.)
Resolution <sup>3</sup>	-	16/24 bit
Other		
	Min	Max
Supply voltage (VIN, -RI* connected)	4.5 V	15 V (absolut max. 20V)
Output voltage (VCC)	-	3.3 V
Baud rate	9600	115200
Internal Sysclock	~3 MHz	49 Mhz

<sup>1</sup> Constant power consumption (including the device itself) shall not exceed 100 mA.

<sup>2</sup> Despite of digital IOs being 5V tolerant analog inputs shall not be driven higher than 3.6V – voltages over 2.5 V are indistinguishable.

<sup>3</sup> Resolution depends on type of device (RS232-ADC16 or RS232-ADC24).

## 8.4 Reaction time

Reaction time depends on two things – the baud rate and system clock of course. Each request has to be received and decoded before any action could occur. The overhead of decoding is partly time-shared with pure transmission overhead thus falling less and less in account in either higher system clock rates or slower transmission speed.

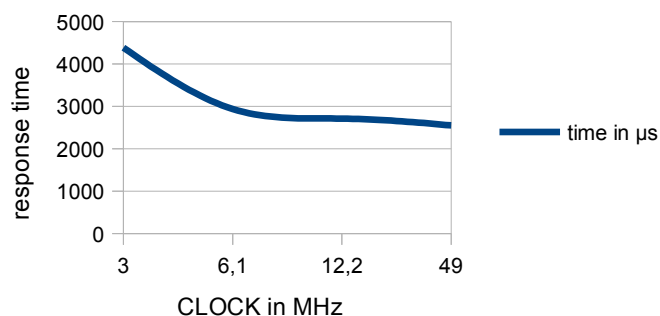
Note that transmission and decoding overhead do not account the main time when making more precise ADC measurements.

The following data considers a scenario where the time between sending a command (e.g. switching a pin's level) and the end of execution (pin has changed to this level) might be critical.

Response time	
$f_{\text{SYSCLK}}$ in MHz	time in $\mu\text{s}$
6.125	4383
12.25	2937
24.5	2711
49	2551

This data is collected at 115200 baud not involving time needed for receiving the request's answer.

CLOCK vs response time



The task was to set the level of  $D_0$  low. Time measurement started when sending the request and stopped when  $D_0$  had become low triggered by interrupt. The length of the request was 14 bytes. According to this, time needed for transmission would be 1215  $\mu\text{s}$  at 115200 baud.

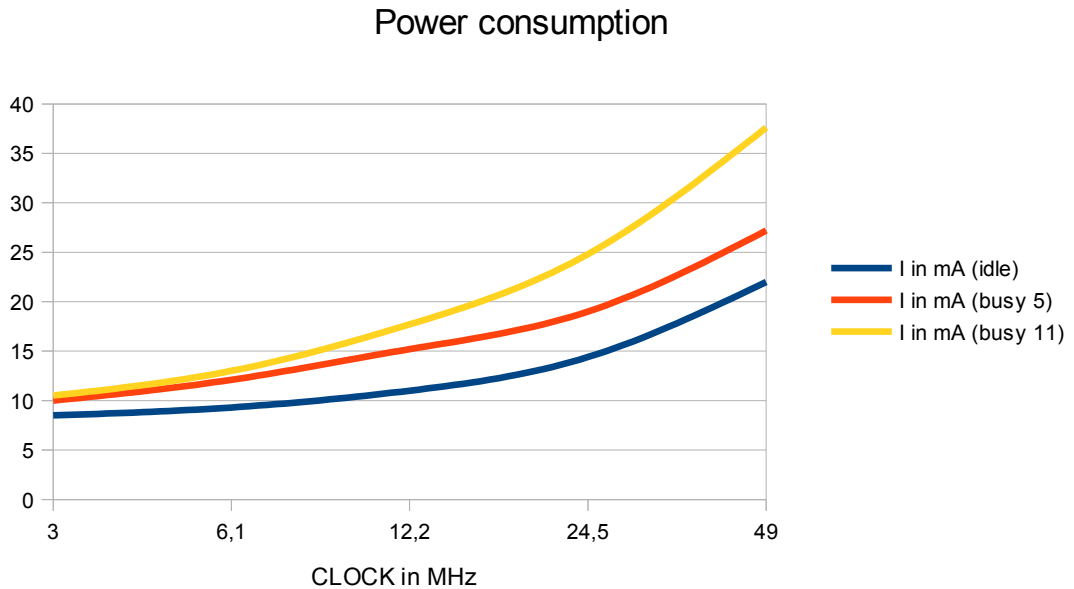
## 8.5 Power consumption

Power consumption depends on SYSCLK settings, ADC decimation ration and the device's busy cycle. If no commands have to be processed idle mode is entered resulting in lower power consumption.

Disrupting the RS232 connection will also lead to lower values.

Power consumption			
$f_{\text{SYSCLK}}$ MHz	Idle mA	Busy (REG_ADC_DEC = 5) mA	Busy (REG_ADC_DEC = 11) mA
3.0625	8.5	10.0	10.5
6.125	9.3	12.1	13.0
12.25	11.0	15.2	17.7
24.5	14.4	19.0	24.8
49	22.0	27.2	37.6

Data is gathered with  $V_{\text{in}} = 5.0 \text{ V}$  using a RS232-ADC16 device.



## 8.6 Baud rate error

Since SYSCLK is not dividable by standard baud rates errors occur when setting transmission speed. The table shows errors for the default value according to SYSCLK.

Baud rate errors			
$f_{\text{SYSCLK}}$ MHz	Desired baud rate	Resulting baud rate	Error in %
3.0625	57600	56179	2.47
6.125	115200	114285	0.79
12.25	115200	114942	0.22
24.5	115200	117647	2.12
49	115200	113636	1.36

## 8.7 ADC

Data of the following tables is collected by RS232-ADC16 device with firmware version 1.11 at various clock and decimation rates.

Precision and performance							
REG_ADC_DEC	I in mA	$f_{\text{output}}$ Hz	Min mV	Max mV	Noise mV	Noise %	Bit
5	19.0	261	-4.73	5.53	10.26	0.4104	7.93
6	19.8	231	-1.03	1.11	2.14	0.0856	10.19
7	20.1	188	-0.21	0.23	0.44	0.0176	12.47
8	22.1	136	-0.15	0.11	0.26	0.0140	13.23
9	23.3	88	-0.12	0.08	0.20	0.0080	13.61
10	24.2	52	-0.08	0.08	0.16	0.0064	13.93
11	24.8	29	-0.08	0.04	0.12	0.0048	14.35

SYSCLK = 24.5 Mhz

Since firmware version 1.11 provides ADC output rates that scale with SYSCLK one might consider the influence of SYSCLK on output rate and precision.

REG_ADC_DEC = 11							
SYSCLK MHz	I in mA	f <sub>output</sub> Hz	Min mV	Max mV	Noise mV	Noise %	Bit
3	10.5	4	-0.04	0.03	0.07	0.0028	15.12
6.1	13.0	7.5	-0.04	0.07	0.11	0.0044	14.47
12.2	17.7	15	-0.04	0.04	0.08	0.0032	14.93
24.5	24.8	29	-0.08	0.04	0.12	0.0048	14.35
49	37.6	29	-0.07	0.08	0.15	0.0060	14.02

REG_ADC_DEC = 5							
SYSCLK MHz	I in mA	f <sub>output</sub> Hz	Min mV	Max mV	Noise mV	Noise %	Bit
3	10.0	76	-4.85	4.84	9.69	0.3876	8.01
6.1	12.1	148	-5.23	4.84	10.07	0.4028	7.96
12.2	15.2	218	-4.81	4.65	9.46	0.3784	8.05
24.5	19.0	231	-4.69	5.38	10.07	0.4028	7.96
49	27.2	261	-4.70	5.53	10.23	0.4092	7.93