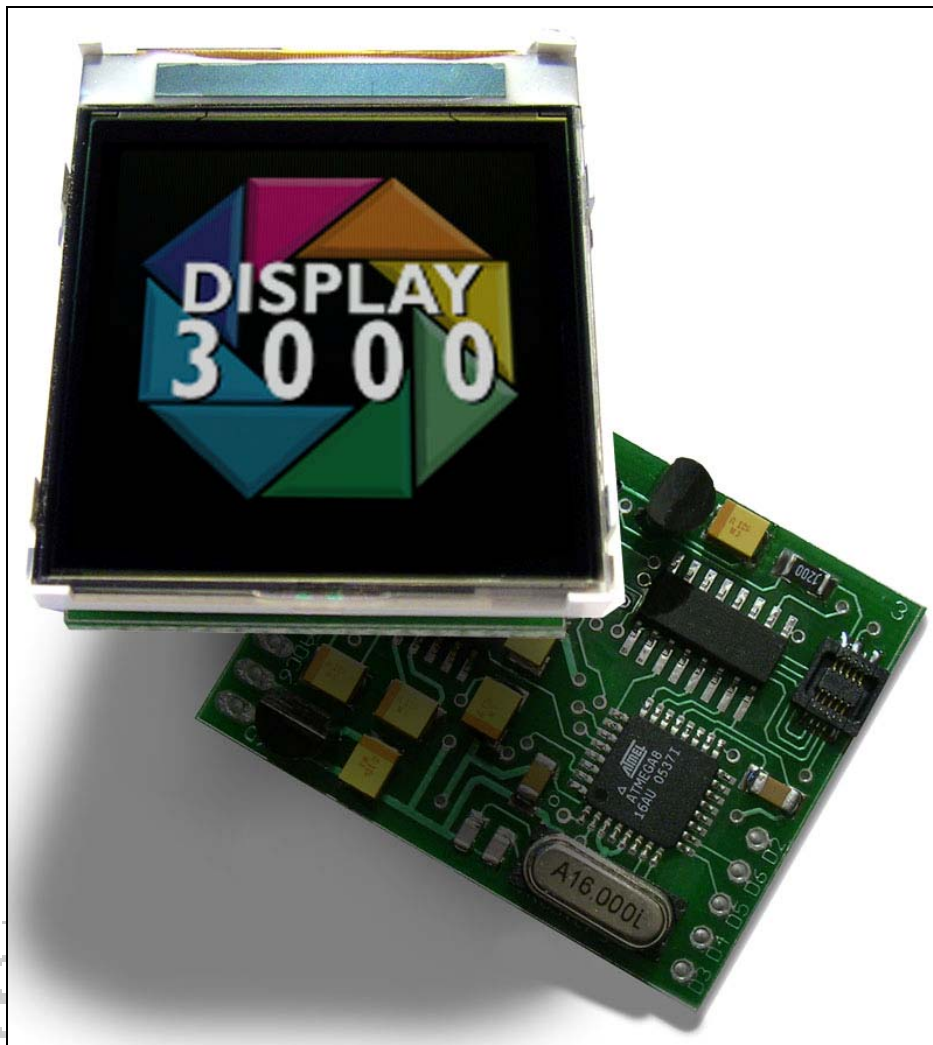


# Supplement for module D04I incl. ATMega8 Prozessor

V 1.4  
16. March 2006



© 2006 by Peter Küsters

**This document is in copyright protected. It is not permitted to change any part of it. It is not permitted to publish it in any way, to make it available as a download or to pass it to other people. Offenses are pursued.**

Congratulations for buying this ATmega8-module.

This module contains the electronics for the color display as well as PowerBooster Technology, a RS232 interface and a ATmega8 processor. This module allows you to keep the complete electronics of your solutions on one board with minimal space.

You need to complete this module by clicking the display into the connector and by soldering your needed cables.

When the module is inevitably frequently moved, this load could damage the patch cord of the LCD in the long run. Therefore we suggest that you fix the display at the lower edge with a drop adhesive or a piece to double-sided tape on the plate.

The display can be removed however only then surely again, if you stick it only at the edge of the plastic. If you are sticking it however on the back, then the back foil of the display is possibly damaged when taking the display off.

Note: the glass of the display or the electronics below may become damaged by a too strong pressing at the display surface.

**This manual shows you just the connection plan of this board and gives you some hints for using it. For the programming information of the color display please check the separate manual for the color display.**

**CAUTION:**

- 1) Never attach the display or remove it, as long as power is switched on
- 2) Connect the display to its connector always correctly (see illustrations on next pages). Never attach differently around! If you wrongly attach the display, it is inevitably destroyed.

**Port usage of this module**

To use this module with your own software you need to know which port of the processor is connected to which display connector. When using our sample software you may need to change the port selection at the beginning as follows:

Display connection	Port at integrated micro controller	Sample programs in Bascom should show at the beginning:
SC	C.0	<code>Const Rs = 2</code>
SD	C.1	<code>Const CS = 3</code>
RS	C.2	<code>Const Sdata = 1</code>
CS	C.3	<code>Const Sclk = 0</code>



## Delivery:

### What you get delivered:

- 1 x PCB with micro processor ATmega 8 etc.
- 1 x color display
- 1 x pins (2x5, 0,1" spacing)
- software, documentation

## Voltage supply

This board is delivered with two voltage regulators to provide you an easy usage and to avoid any damage to the processor and the display.

You may run this module with any DC voltage from 5 up to 20 Volt. The processor and the RS232 part is running with 5 Volt, the display electronics with 3 Volt. These voltage regulators are very-low-drop-regulators so you really can offer 5,0 Volt as a minimum to the board (and not approx. 6,5 V as a minimum as you would need with a usual 7805 regulator).

### Caution:

The 5 Volt regulator is able to deliver up to 100mA current – the board itself needs about 50 mA as the 8 Volt for the display lighting is being produced at the board too. Please do not draw too much current from this board for any other items you are planning to connect to this board, you just have approx. 50mA left.

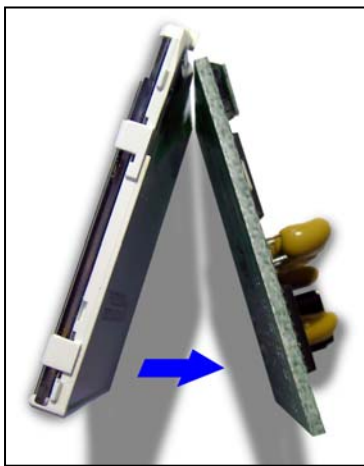
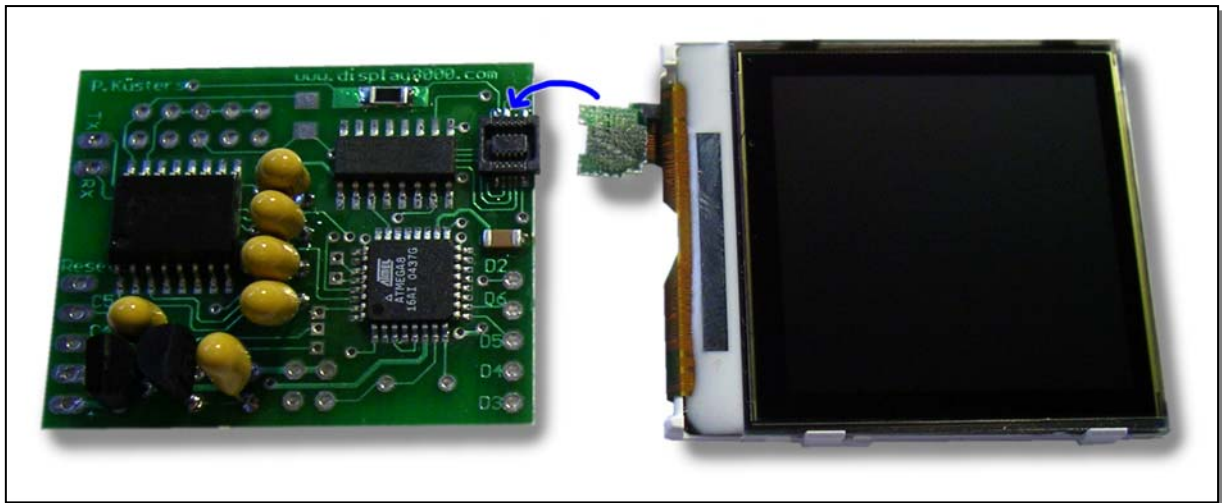
If you need more current, it would be a good idea to bridge the internal 5 Volt regulator (the one directly beneath the Vcc pad) and to use an external regulator which may provide larger current.

But then never provide a larger voltage to the board than 5,0 Volt. A higher voltage may destroy the display lighting and/or the processor on the board.

## How to get it up and running

Please lay display and PCB on a table as shown below. Then click the connector into its place and carefully fold the display to the back of the PCB.

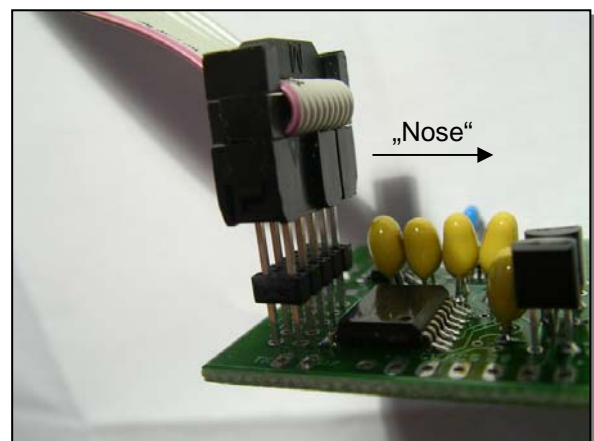
To disconnect please do the same in the reverse order: First unfold the module, then unclip the connector (if your PCB looks different: newer modules have a different PCB layout but the connection of the display remains the same).



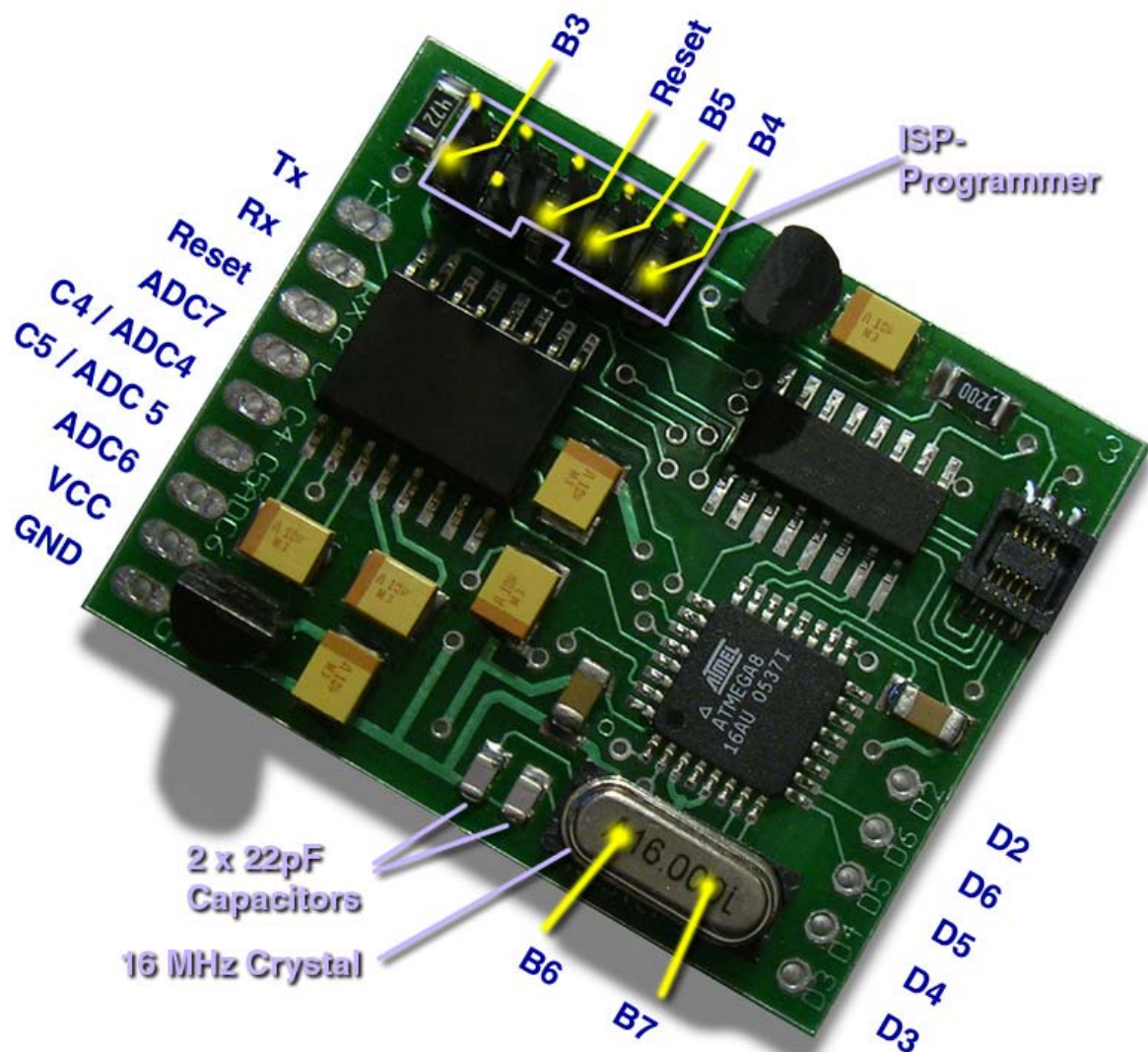
**HINT:** You may use double sided tape to fix the display to the back of the PCB. This would be an easy and powerful fixation. But please keep in mind: You will never be able to dismount the display then: if you try to pull it from the tape, you will damage the foil of the display back and therefore destroy it.

Now provide a voltage of 5 up to 20 Volts to the pads Vcc and GND. We already programmed the processor so you will see a program showing you the status of the switches and the usable ports.

You need a separate ISP-programmer to be able to reprogram this board. As of the limited space on this board we were not able to leave room for the usual black connectors with the plastic frame around it (if somebody knows the English name for this special connector: your suggestions are welcome). As for this reason you need to use the provided pins without the plastic frame. AS the frame usually prevents a wrong connection, you now need to take care of this by yourself: **The “nose” of your ISP cable always has to face the chip on the PCB (see picture)**



## The pads of the module



The photo above shows you the available pads of the board. The description of the pads is also shown on the PCB itself.

Because of the very limited space of the PCB we were not be able to offer you pads of all ports of the micro controller. The available are D2, D3, D4, D5, D6 and with C4 and C5 two ports which may be used as digital input/output or analogue input ports. Additionally there are two analog inputs with ADC6 and ADC7.

If you do not want to use a crystal, you may also use Port B6 and B7. Whenever you solder a crystal to the board (and change the fuse to external crystal) these two ports are not usable anymore.

You may also use B3, B4 and B5 of the ISP port. This result in **14 available ports** which may be used as input or output, four of them may be used to measure analogue signals. Also there is a pad for using a reset-switch – just connect Reset to GND to trigger a Reset at the micro controller - this starts your program from the beginning.

The micro controller actually runs at 8 MHz. You may double the speed by soldering a 16 MHz crystal and 2 x 22pF ceramic capacities to the board. You then need to reprogram the speed fuse from 0100 (8Mhz) to 1111 (external crystal 16 MHz).

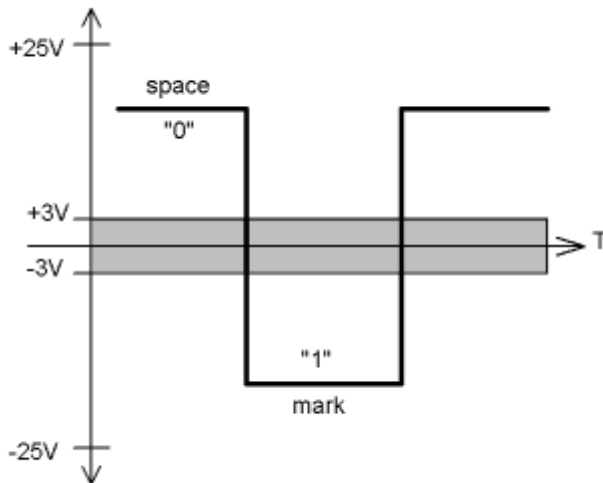
**Caution:** any other selection than 0001, 0100 and 1111 **may cause a malfunction** of your board!! It will also not work anymore if you select 1111 without having a crystal soldered in. **Fuses are very delicate** – if you are not experienced with them, **do not experiment with changes – quickly the board will not function anymore!!**

**WARNING:** There is a fuse available, which allows you to change the Reset-Pin of the controller to a port (C6). Whenever you do this, you can not return this step. The ISP programmer needs Reset for a correct working. When you change the fuse, you will have one port more available, but you will never be able to reprogram your module or the reset the fuse to its original state. You will need to solder a new controller in if you want to use the board furthermore.

## RS232

RS-232 is a serial communication protocol. It sends information as bit after bit and has two signal levels:

- a voltage between -3 and -25 Volts is a logic one (1)
- a voltage between +3 and +25 Volts is a logic zero (0)



As the picture above shows, the voltage level between -3 and +3 Volts is undefined. In practice this is not so. Most often, any voltage level above 2.5 Volts is seen as a logic zero, anything below as a logic one.

The electrical specification of RS-232 is quite robust, all outputs must be able to sustain a full short-circuit and all inputs must have a schmitt-trigger action. This makes a full-standard RS232 port on a PC much less vulnerable than a TTL-level parallel port.

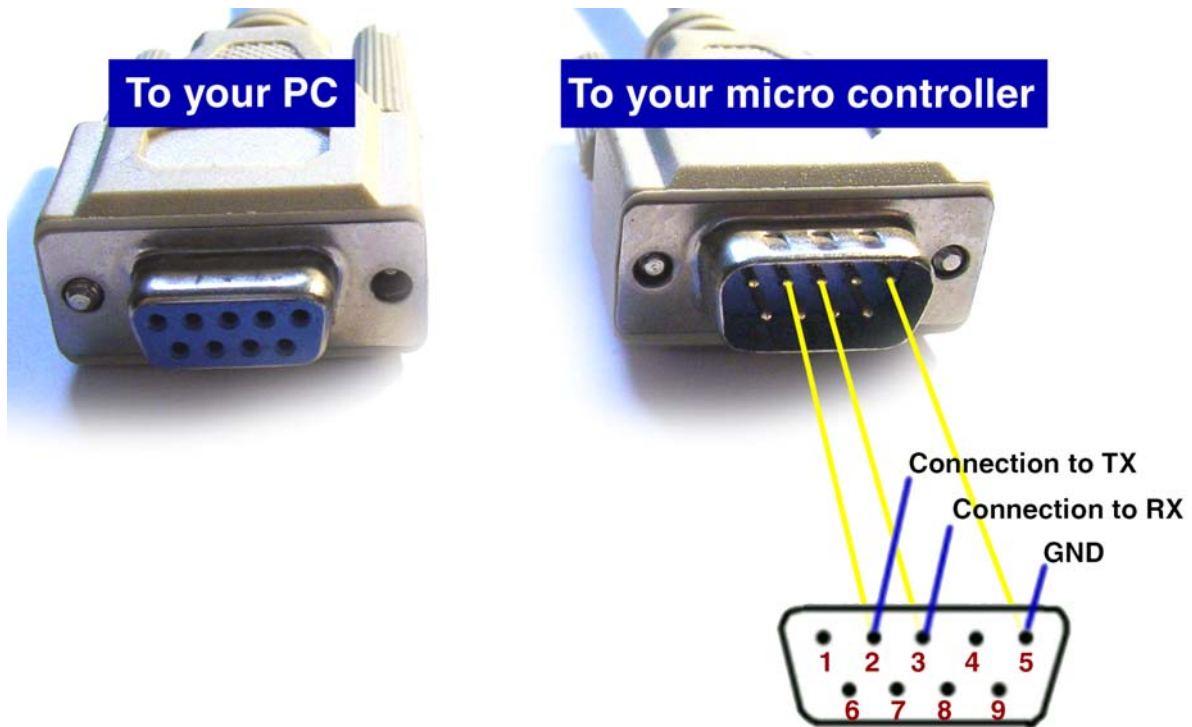
RS-232 is an a-synchronous protocol, meaning that no separate clock is transmitted with the data. Both sides must know the communication speed (we use the term baud-rate) beforehand.

RS-232 usually defines a complete hardware handshaking system using several wiring pins. We use only the most important three:

- RxD : receive data, pin number 2
- TxD : transmit data, pin number 3
- Ground, pin number 5

These pin numbers refer to a standard male DB9 connector on your PC or laptop.

If you want to build a cable to connect our board to your PC you need a regular serial cable with one male and one female DB9-connector. The female will be connected to your PC, the male needs to become connected to you board. The following picture will help you to build a adapter and to connect the cable to your board.



Pin 2 is the receive channel of the PC –you need to connect this channel to the transmit channel (TX) of the micro controller. Also at Pin 3 the data of the PC are transmitted to our board and therefore you need to connect Pin 3 with the receive channel (RX) of the board.

The ATmega 8 offers one RS232 interfaces. At the ports D0 and D1 the used RS232 interface of the ATmega 8 is located. These two ports (D.0 and D.1) are connected to the RS232 interface chip at the board and this chip is connected to the Rx and Tx Pad. The chip decouples the high voltage RS232 signals from your PC. If you would connect the PC directly with the ATmega, the micro controller would become destroyed.

If you want to use the RS232 interface, the following example might be helpful for you. Using the interface is also helpful during debugging of your code, as you just “print” variable values to the interface and check at the terminal program of your connected PC if the variables contain what you expect. At MS Windows<sup>®</sup> you may use either the Hyperterminal<sup>®</sup> which comes with MS Windows<sup>®</sup> or with Bascom<sup>®</sup> you may use the internal monitor for this. You may use the following program to test the output of your module and the terminal program of your PC.

```

`sample program RS232 output
$regfile = "m8def.dat"
$crystal = 8000000
$baud1 = 9600

Do
  Print "Hello world"
  Wait 1
Loop
End

```

### RS232 and the crystal / Overclocking the board

If you are planning of sending a lot of information through the RS232 interface you need to know, that the frequency for the selected baud rate is calculated by the micro controller using the current clock rate. Two facts are important to know:

a) The internal resonator is not very accurate and varies with different temperatures etc. So if the board runs with internal 8 MHz, using of the RS232 interface may result in transmission problems. You better use an external crystal then.

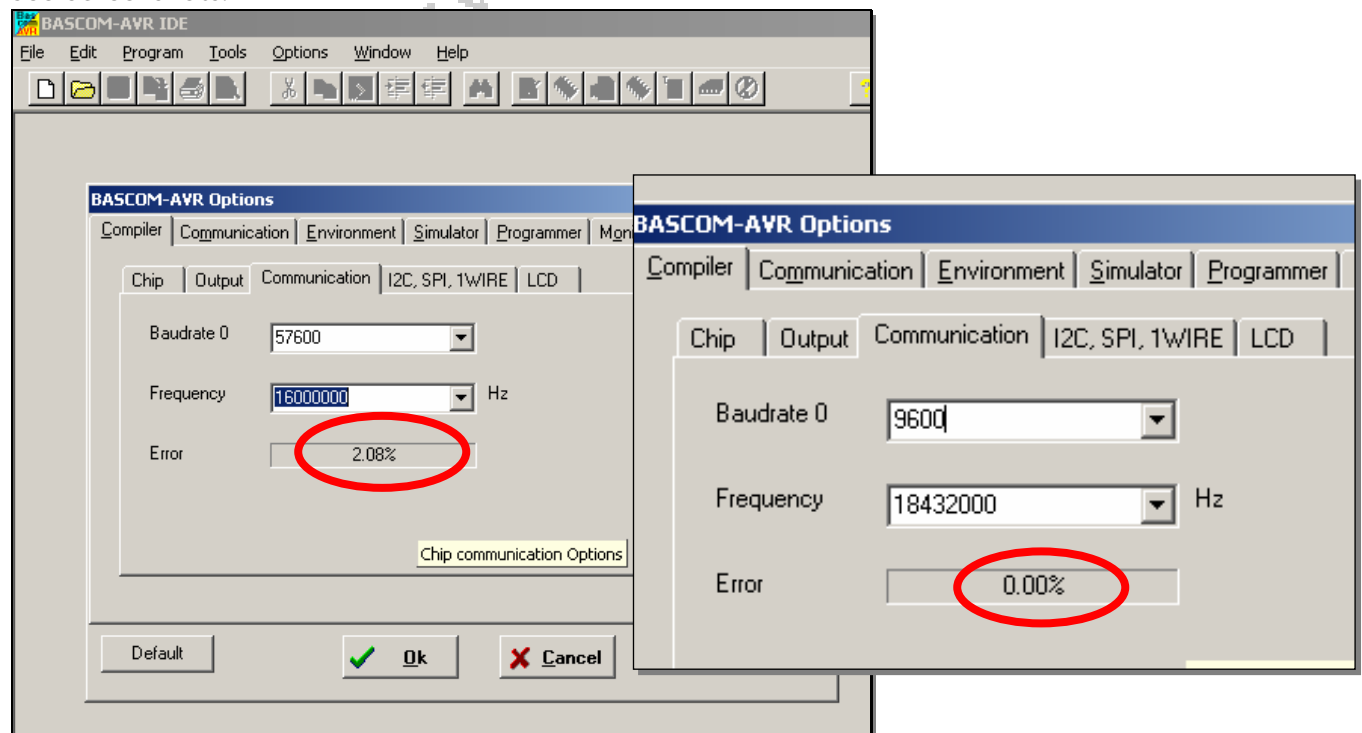
b) Using the usual 16 MHz crystal as the external clock will result in a not 100% correct frequency of the RS232 interface – varying by the selected baud rate. The perfect match would be a crystal of 14.7456 MHz or 18.432 MHz instead, as they will result in 100% correct frequency. At 14.7456 the micro controller is a bit slower, with 18.432 you are overclocking the micro controller. Usually this will not result in problems as the ATMega128 can easily run at a even higher clock rate. The internal EEprom is the area which will first show errors during overclocking – you will not be able to read or write correctly to it. If you do not need the Eeprom you may run the board even at 20 MHz.

c) You need to tell the controller, what clock frequency you are providing – otherwise the wrong calculation is done and the transmission will not work. In Bascom® you do this with the command `$crystal = 8000000` at the beginning (8000000 for 8 MHz; 16000000 for 16 MHz, 14745600 for 14.7456 MHz etc.).

You always need to enter the exact speed of your crystal – do not enter any different value, as this will cause in a wrong timing.

At Bascom®, there is a calculator included, which baudrate is possible with the selected frequency. You will find this at the menu **Options / Compiler / Communications**.

See Screenshots:



This will help you to check if you get an error free transmission with your crystal.

**In short:**

As the internal 8 Mhz is not accurate you should always solder an external crystal in if you want to use RS232.

16 MHz creates usually 0,16% error rate at slower speeds which is OK. Usually an error rate of less than 1% should not result in a higher error rate during transmission.

14.7456 MHz or 18.432 MHz will result in 0,00% error rate and is the perfect selection when using RS232.

(c) 2005 www.display3000.com

## Technical data of display module kits:

### Item D041:

<b>Size:</b>	40 x 35 mm (1.6" x 1.4") ca. 15 mm height (with display)
<b>Voltage:</b>	5 to 20 Volt DC, approx. 50mA
<b>Processor:</b>	ATMega 8 8 KByte flash 1 KByte RAM 512 Byte Eeprom max. 16 MHz clock (we deliver set to 8 MHz)
<b>Display:</b>	132x132 Pixel, 65.536 colors Active diagonal size: 1,5" (38mm)

We did reprogram the processor – it is not using the standard like ATMEL is delivering the processor. This is what we changed:

**Fusebit:** Speed from 0001 (1MHz intern) to **0100 (8 MHz intern)**

**Fusebit:** **Preserve EEPROM** when chip erase

We programmed EEprom and flash for testing purposes (item is delivered incl. installed testing program)

### Manufacturer:

Speed IT up  
Owner Peter Küsters  
Wekeln 39  
47877 Willich  
Germany  
Telephone: +49-21 54-88 27 5-0  
Fax: +49-21 54-88 27 5-22

Further information and updates: [www.display3000.com](http://www.display3000.com)

Author of this manual: Peter Kuesters  
© Peter Kuesters