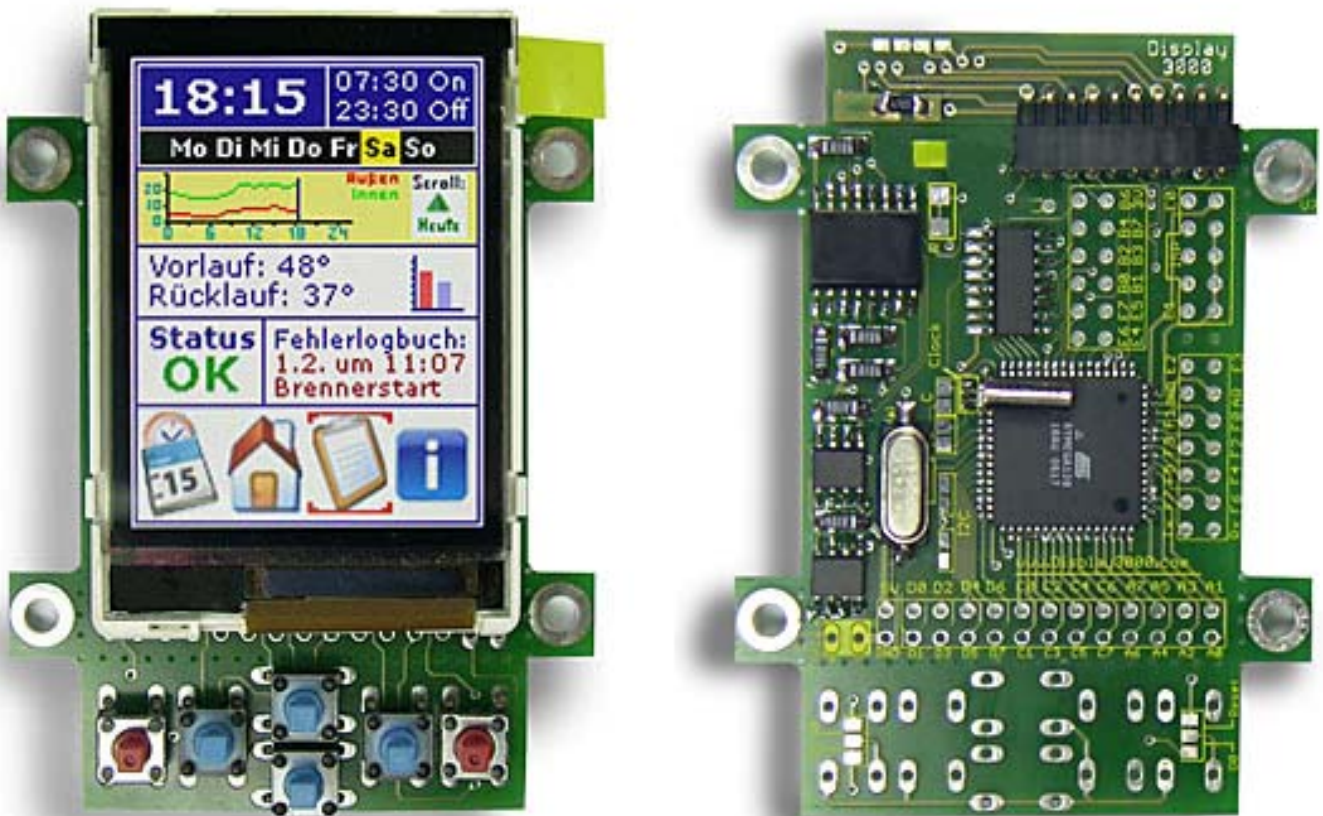


Supplement (hardware manual) for module **D072x**

with Atmel Mikrocontroller:
ATMega128 or **ATMega2561** or **AT90CAN128**

Version 1.9 (for Board D072 – V10)
2. October 2008



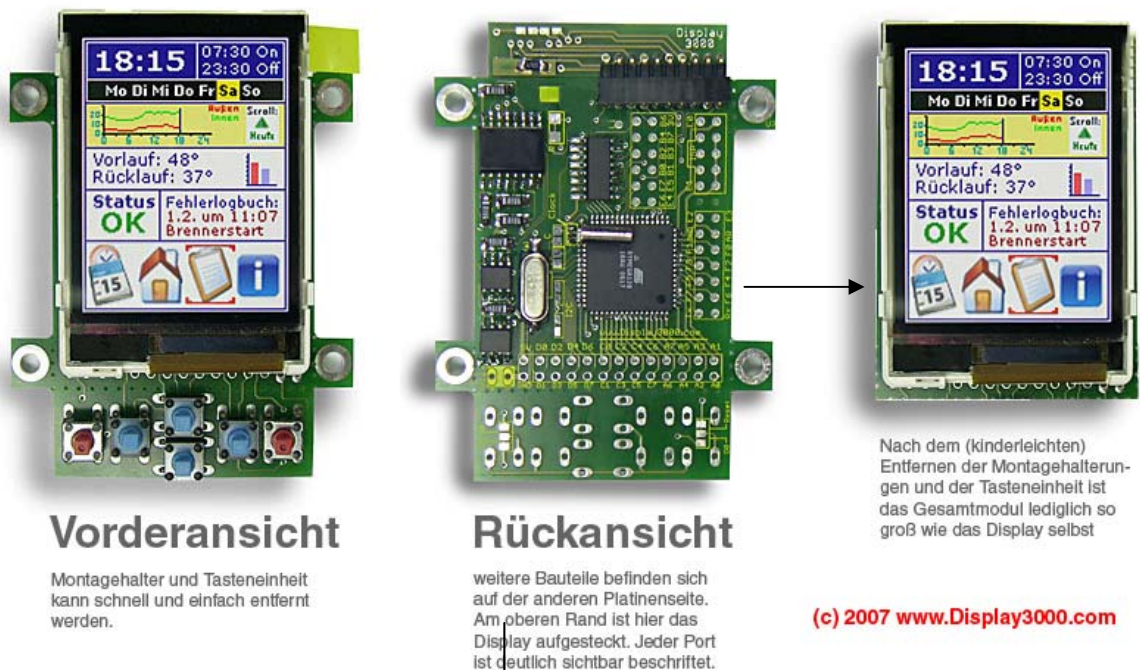
© 2008 by Peter Küsters

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 D072 microcontroller module including color graphics TFT.

This module contains the microcontroller (depending on what you ordered a ATmega128, ATmega2561 or AT90CAN128), the electronics for the color display as well as Power-Booster technology, a RS232 interface, a I2C compatible interface. This module allows you to keep the complete electronics of your solutions on one board with minimal space. If you imagine, this module might have been able to run a moon rocket 30 years ago OK, lets say two modules would be able to run it ☺

At the left and right side of the module you will recognize our mounting pads for mounting the module. These pads can be easily removed by you. The same applies to the switch area below the display. The needed steps to remove frame or switches are shown starting from page 36.



You need to complete this module by clicking the display into the connector and by soldering your needed cables and eventually the switches. We did not soldering the delivered pins, as you might want to use a different kind of connection. Also the switches are not connected yet of the same reason. You may just push them into place now and they will work (for testing purposes) without soldering as of the through plated holes.

By the way – handle the TFT with care. If you take your computer monitor in front of you (which might be a TFT) and throw it from your table, it will, very likely, be broken. The same happens with our small TFT. If you drop it or if you press to hard on its surface it will break.

This manual shows you the needed hardware information of this board and gives you some hints for using it.

For the programming information of the color display please refer to our separate programming manual which is placed at the CD.

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.

New in version V10 (from V9):

- CAN Transceiver already on PCB if you ordered a module with AT90CAN128.
- The two, so far unused Pads beneath the ISP connector are now used for CAN L and CAN H (if a AT90CAN128 is present, otherwise these pads are unused)
- Individual configurable switch off options for saving energy
- The RS232 Transceiver is now not connected to D2 and D3 as default. To use the RS232 transceiver, you need to close two solder bridges with a drop of solder.

New in version V9 (from V8)

- Switch off mode for saving energy
- The inner pads (Port B and E) will now show 5V, 3V and GND as default. This allows an easier usage of external modules. The board can be reconfigured (solder bridges) for compatibility reasons to offer the same ports there as before (as B4 and B6 were also used by the display, it is very unlikely that somebody used them)

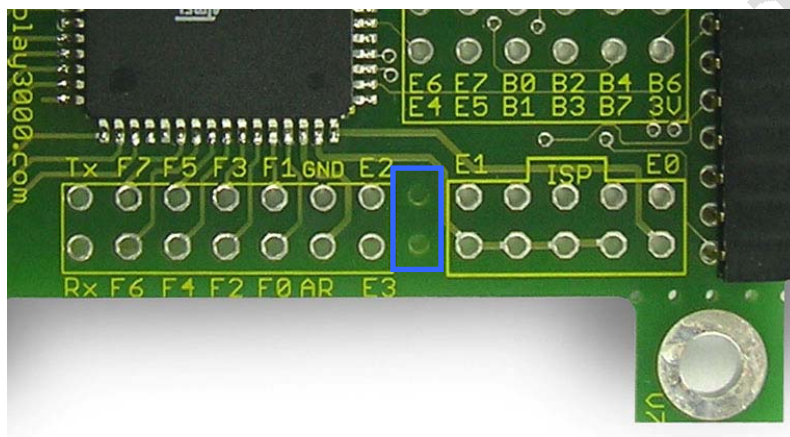
Delivery:

What you get delivered:

- 1 x PCB with micro processor ATmega 128 etc.
- 1 x TFT color display with connector PCB
- 6 switches
- connector bars (0,1" spacing)
- CD with sample software, documentation, utility software

First solder the connector bars to your PCB. Alternatively you may solder different types of connectors or even some wires to it of course. If you want to use this module for experimental usage you better think of purchasing our experimental board P006. This allows you to solder every external device to the extra PCB and then just plug the D072 into it without any soldering at the main module. Even after lots of experimenting and soldering your module looks like new.

Important: Before soldering the connectors to the board please read the following note.



If you, as shown on page 11, want to connect the ISP connector directly with this module, you better split the connectors into two pieces with 2x5 and a 2x7 connectors (or you pull the two contact pins out of the plastic frame with pliers). The holes marked by the blue frame shall be unused as otherwise there would be not enough room left for the ISP connector.

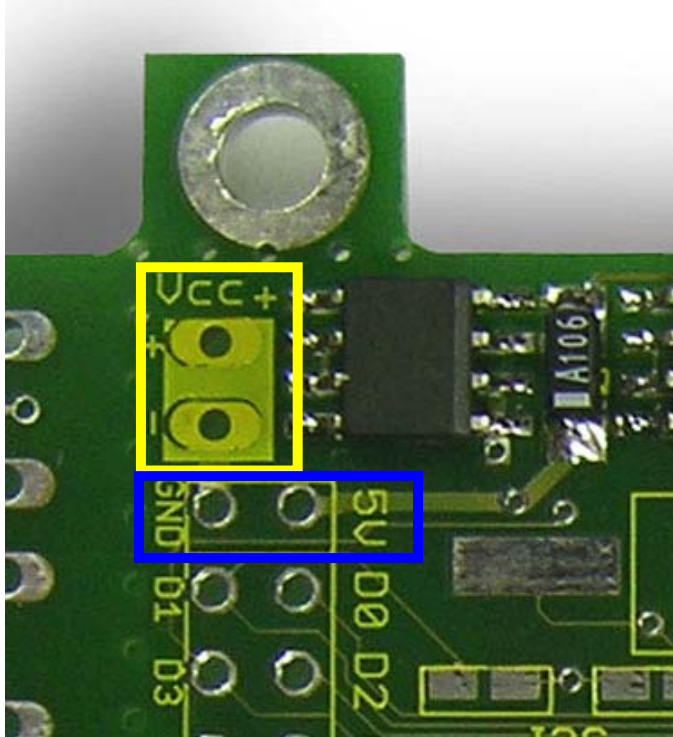
tor (the plastic frame around the connector) – you will then not be able to connect it to the pins then.

If you use one of our experimental PCB such as P005 or P006 this precaution is not needed and you may add all pins.

New in version V10: until now the mentioned two pads were unused. Now with V10 these pins (blue frame) are now CAN L and CAN H. This only concerns you if you ordered a D072 with a AT90CAN128 microcontroller and CAN transceiver. Otherwise these two pads are still unused.

Voltage supply

This board is delivered with complex 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 4,5 up to 20 Volt (yellow frame). 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 4,5 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). If you want to use the internal reference voltage for using the analog-digital converters of the ATmega, you should connect the module with at least 6 Volt to get a stable 5 Volt voltage.

Caution:

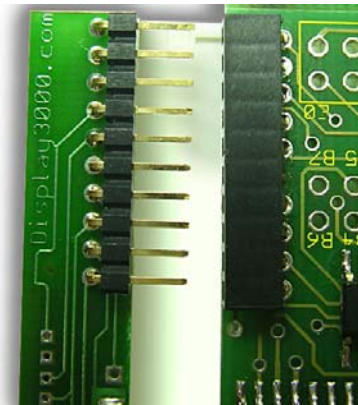
The 5 Volt regulator is able to deliver up to 160mA current – the board itself needs about 50 mA as the 8 Volt for the display lighting is being produced at the board too. Please do not draw too much current from this board for any other items you are planning to connect to this board, you just have approx. 100mA 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. Alternative: offer 5 Volt or less directly at the pads 5V and GND (blue frame). 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

Connect the display and provide a voltage of 5 up to 14 Volts to the pads Vcc and GND. The display should be mounted like shown at the following pictures:

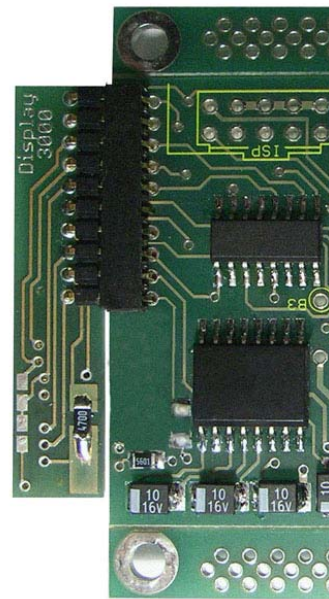
1) Place the display on the table in front of you. Now place the module in a way, that module and display connector will face each other.

Remark: if you look at the tiny display PCB you will recognize the special SMD connector inside. Due to a larger stress (e.g. dropping the display), this connector might spring open. To prevent this connector to open accidentally, you shall fix it with a drop of glue at the right side of the display (left side has already been fixed by us).



2) Hold the connector PCB of the display with two fingers and push the connector gently to towards the connector on the main PCB.

Always make sure that you will not mistakenly connect the display wrong – it might become destroyed when you power up the module.



3) Ready

This connector gives you the possibility to remove the display if you want and to mount it in distance from the main PCB. Then, the cable should not be too long. 20-30 cm (10 inch) were tested by us and should work.

If you want to connect or disconnect the display, always switch off power before.

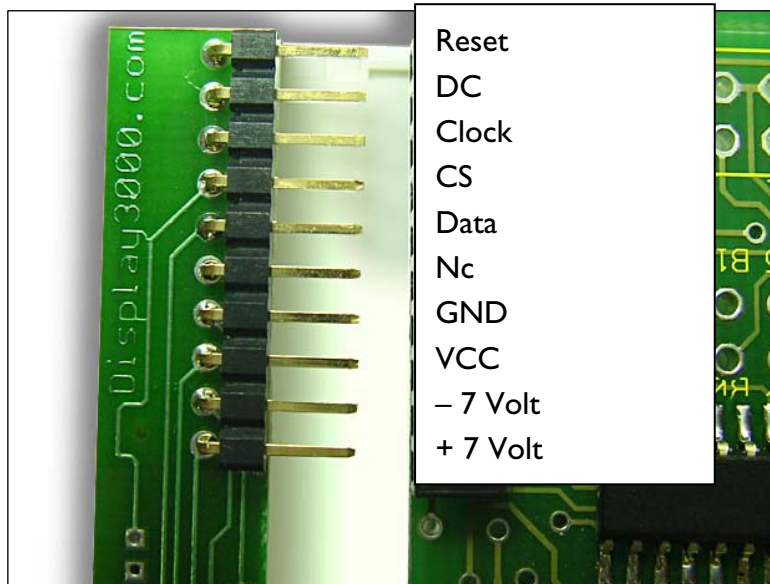
We already pre-programmed the controller so you will see a program showing you the status of each single port. The ports F4 to F7 show „0“ as they are prepared for the JTAG interface and are not usable otherwise until JTAG is disabled (see last page on details).

Programming interface

You need a separate ISP-programmer to be able to reprogram this board. This programmer is connected to the D072 board and your PC.

The connection of the display

The separate programmers' manual on our CD explains all details how to drive the display, how to use our software and how to use our utilities. The connector scheme of the bare display is showed below.

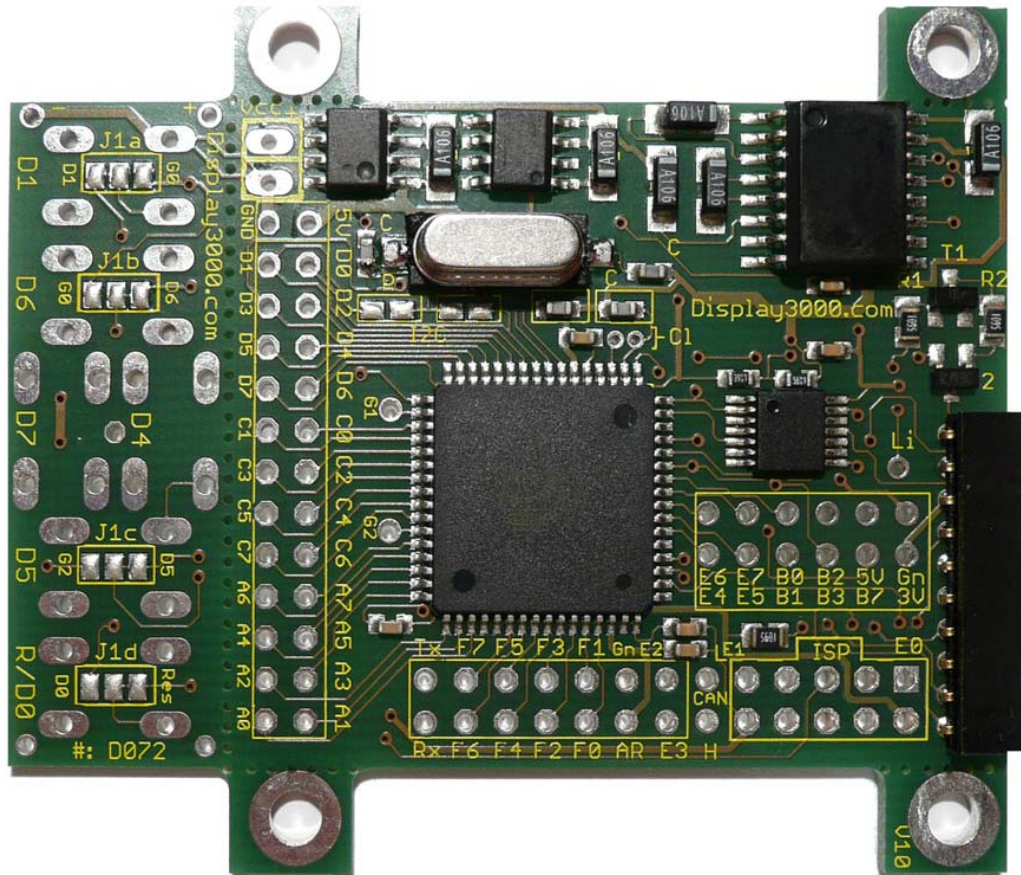


This module offers some special options which we also explain in the upcoming chapters of this manual. **If you did not order these options from us, you need to upgrade your board by yourself before you are able to use these.**

1. Speed up your module by adding a SMD crystal
2. Add a clock crystal for the internal RTC (Real Time Clock)
3. Switch the display lighting on and off by a switch
4. Switch the display lighting on and off by the controller
5. Dim the display lighting by the controller
6. Usage of the TWI-Interface of the controllers (e.g. connection of I²C devices)
7. Usage of the CAN-Bus with a AT90CAN128
8. Energy saving mode (switch off individual configurable parts of the module)

The pads of the module D072

You received a board including the printing information of each pad.



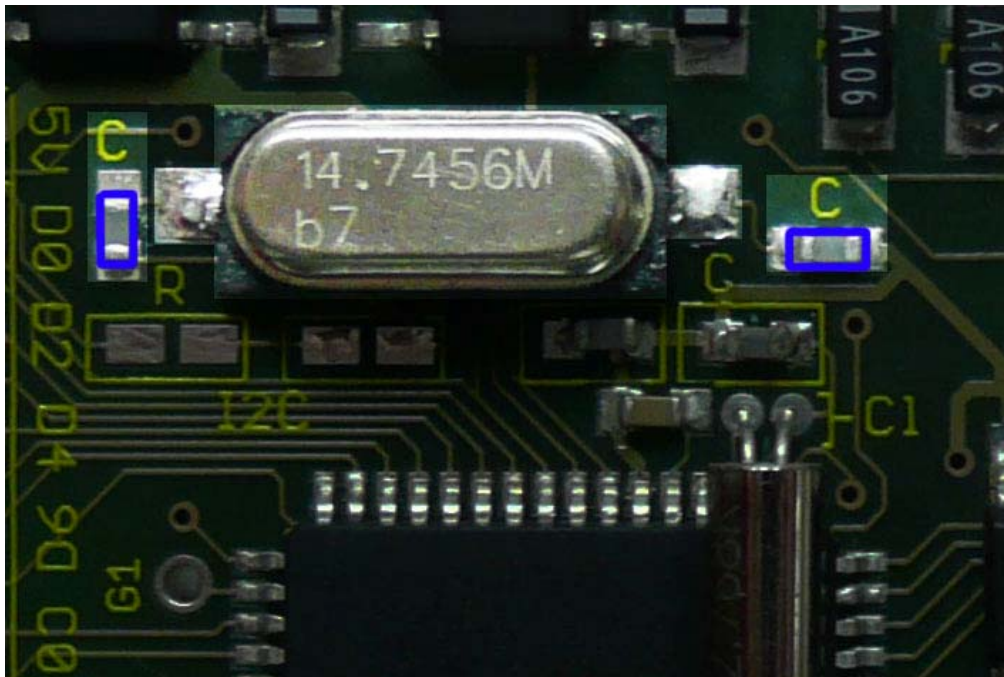
There are also at front and back of the PCB soldering bridges for several options. Each soldering bridge is marked with a number which are described in this manual. At the end of this manual you will find a summarization of all bridges.

One aspect upfront:

For the usage of the **RS232 Interface** you need to close two soldering bridges. Please check page 29.

Adding a clock crystal

If you ordered a board without an additional crystal and if you want to solder a crystal in by yourself, you need also to solder 2 capacities in. These are the shown 2 x 22pF capacities at page 8 (located below the crystal at the opposite side of the PCB). The crystal can easily being located at the following picture. Please make sure you buy a SMD crystal and SMD capacities (size 603 – caution: this are VERY tiny parts).



By default, without an ordered crystal, the micro controller at our board actually runs at 8 MHz with an internal resonator. You may enhance the speed by soldering a crystal and 2 x 22pF ceramic capacities to the board. You then need to reprogram the speed fuse from 0100 (8Mhz internal) to 1111 (external crystal).

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!!

Overclocking:

If you select 1111 as the fuse value, you may solder any crystal speed you want. 16 MHz is the official limitation of the ATmega128, however it can sometimes be overclocked up to 20 MHz.

But be aware: The slowest part of the micro controller is the build-in Eeprom. While a 10% higher frequency is usually no problem, 20 MHz will often result in wrong data if you write or read from the Eeprom. If you do not use the Eeprom you can use higher frequencies. Any overclocking is at your own risk. If you plan to build critical applications – do not overclock!

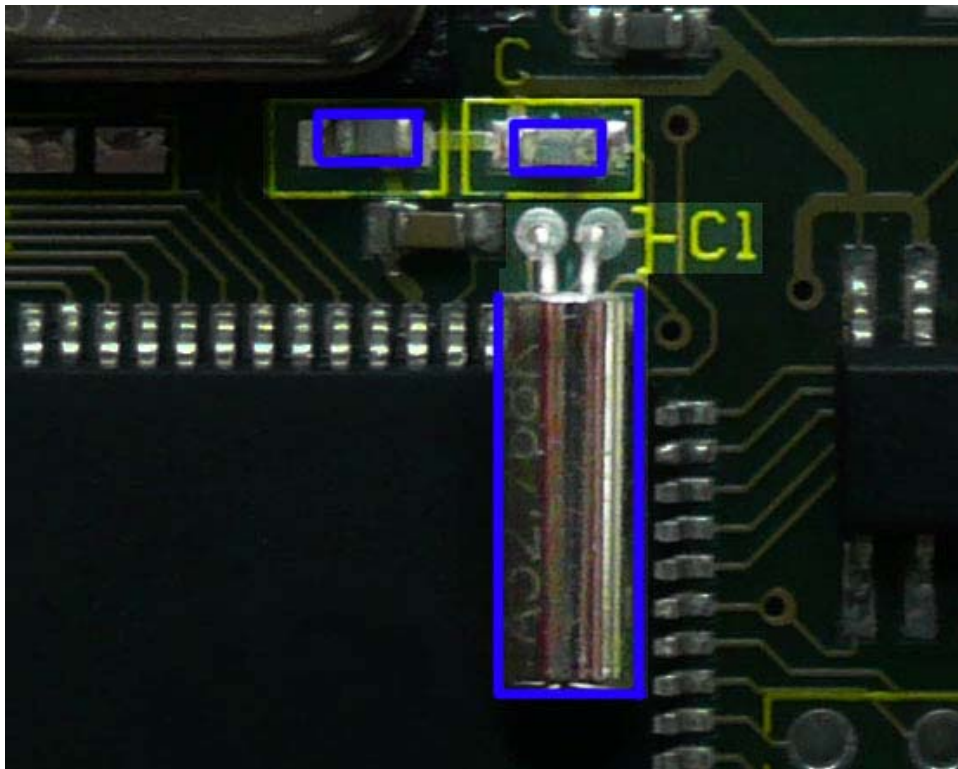
The selection of the correct crystal is also depending on the general speed you need (by the way: the faster the ATmega runs, the more current it needs), but also on the need of a RS232 interface. For this, please read the RS232 section at page 24.

Adding a clock crystal for using the module for exact timing

We are prepared the board for the usual 32.768 KHz clock crystal. You should use this if you want to use an exact timer, like running a clock with your module.

This is also useful if you want to save battery power: you may send the ATmega to the sleep mode and wake it up e.g. once or twice a second, let it check some status and return to sleep if no action is needed. By this your microcontroller would sleep 99% of the time.

Hint: Bend the clock crystal after soldering by 90 degrees, so it may lie directly on top of the ATmega to get it out of the way.



Special case: ATmega256I and a clock crystal:

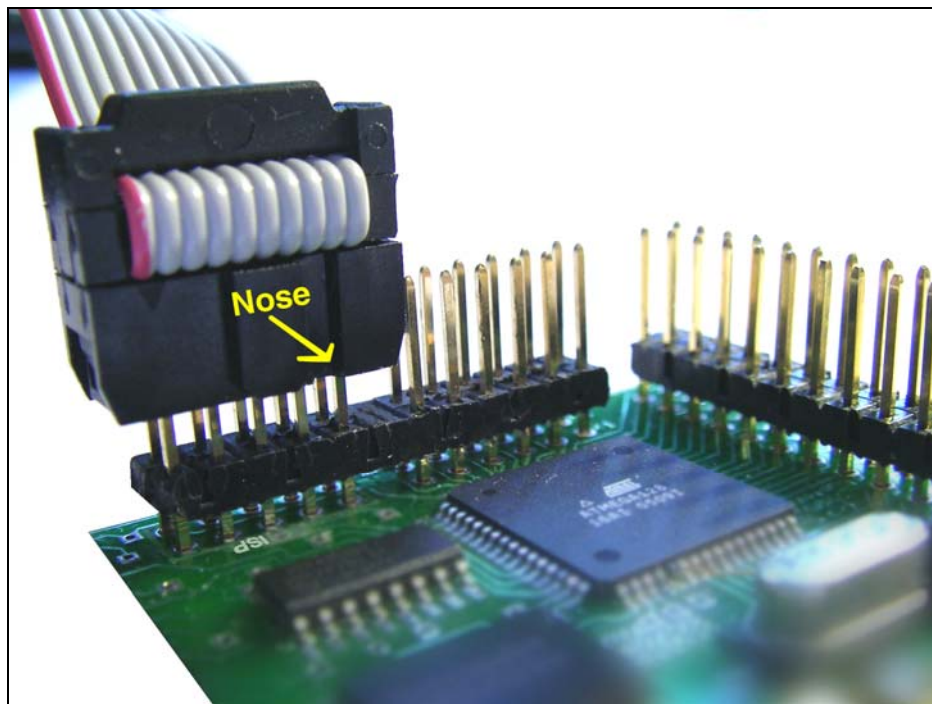
According to its datasheet, the ATmega256I needs extra capacities of 22 pF when a clock crystal is being used. For this reason we prepared the PCB: Directly above the two pads for the clock crystal there are 4 soldering pads for two SMD capacities (22pF, SMD size 603). You only need to add them when using a ATmega256I. If you order our board including clock crystal these are already soldered in.

The ISP-conconnector

As we tried to reduce the needed space as much as possible, there is no room for the standard-ISP connector including his plastic frame. Instead we do offer pins with the same spacing and contact scheme as the regular ISP connector.

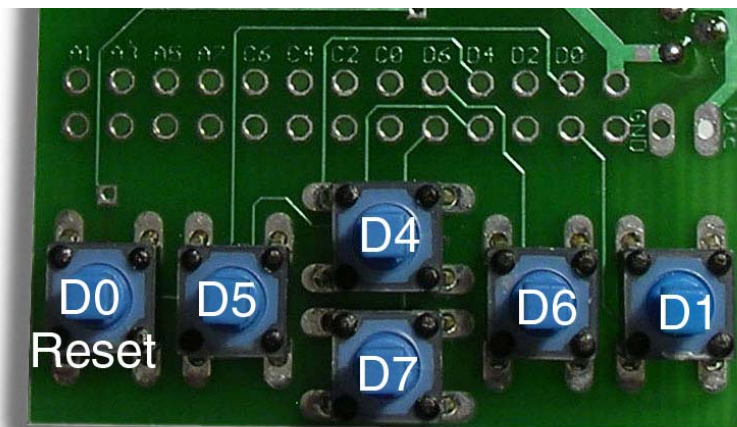
Please refer to the following picture, how you need to connect your ISP cable to our board. It is easy: just remember that the “nose” of the cable has to show towards the chips.

For making remembering easier, there is “ISP” printed on the board as a hint.

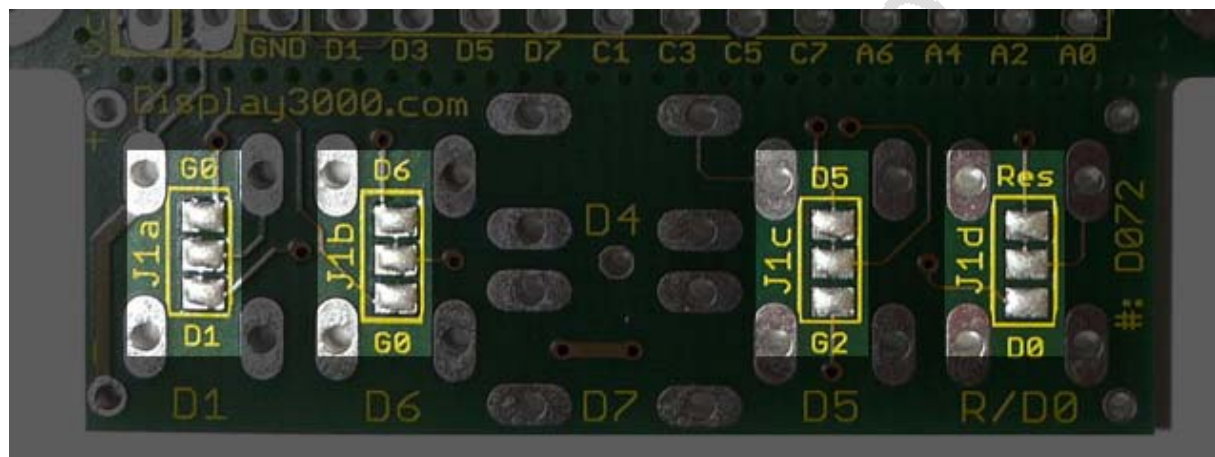


Do not try to connect to the ISP from the display-side. Always connect from the micro controller side as shown in the picture above.

The switches of the D072



The switches of module D072 are already connected to Port D (see photo left). The connection of the 2 outer left and the 2 outer right switches can be changed by soldering bridges (J1a, J1b, J1c, J1d).



The (view from the display side) right switch is connected to port D1 by default. As the ATmega allows multiple usage of each port, this port D.1 does offer a multiple usage too: it might be the data line of a I²C bus system for example. If you want to connect your module to a I²C bus, you probably want to use a different port for this switch. We prepared the board so you may use port G.0 instead for the right switch.

If you would like to change the connection of a switch, you need to cut an existing bridge with the current connection (middle and lower pad) and then close another bridge for the new connection with some solder (middle and upper pad).

Similar is valid for the other three switches. The 2nd switch from the left can also be switched from D.6 to G.0, the 2nd key from the right can be D.5 or G.2 and the outer right switch can be Reset or D.0.

Switch off option for saving energy

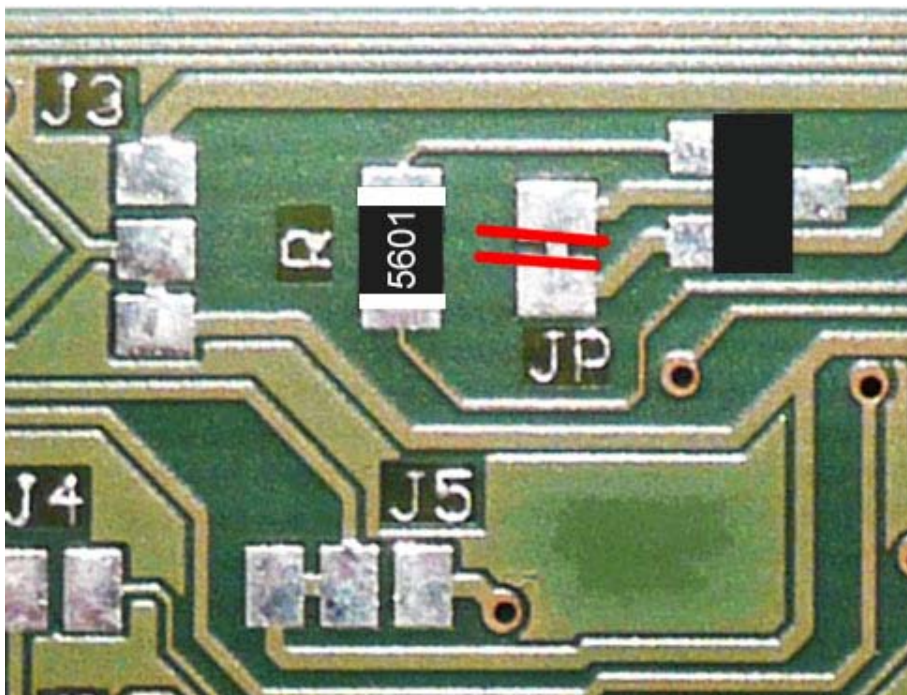
There are situations where saving energy is needed and where the microcontroller can be set to standby. As long as all the other peripherals are still running and consuming power, the saved energy is limited. Especially with a battery driven device this is not perfect. For this you can order the option to switch off several devices by the microcontroller.

By a jumper (soldering bridge) you may select for each individual device if you want to switch it off by a command of the microcontroller.

These devices are:

- 3 Volt voltage regulator with display and display electronics (but not the display lighting) (Jumper J11)
- RS232 Transceiver, Booster for the display lighting (Jumper J3)
- CAN Transceiver (only with a AT90CAN128 microcontroller) (Jumper J2)

To add this option later by yourself you need to solder at the shown positions a transistor (BC 857A) and a 5,6K resistor – both SMD. By cutting the jumper field JP (closed by default) you now can disconnect the devices mentioned above from Vcc. The transistor switches them on or off.



With this, the needed current can be limited to a minimum. After switching everything off it now is a matter of the configuration of the microcontroller and your software how much current your board needs.

The different power modes are described in the datasheet of the microcontroller at section „Power Management and Sleep Modes“. There, everything useful is explained (e.g. switch off ADC, switch off pullup resistors etc.).

If you switched off the display, you need to re-initialize it first before it works (LCD_Init)

After opening jumper JP you can switch off/on everything with G1.

Port G.1 = 0 Switch on

Port G.1 = 1 = Switch off

By default, the devices on the board are switched on permanently. With Jumper 11, Jumper 3, Jumper 2 you can reconfigure them to be switched on or off. (see also the last pages and page 29).

Switching the display lighting

Especially with battery driven devices the current usage of the display lighting might become a problem. Having a possibility for switching it off would be nice. As the lighting LEDs does have a life span of about 10.000 to 20.000 hours (which means 1 to 2 years continuous run) this might also be important for devices which will run for years (remark: life span means: the brightness of the lighting went down to 50% of the initial brightness).

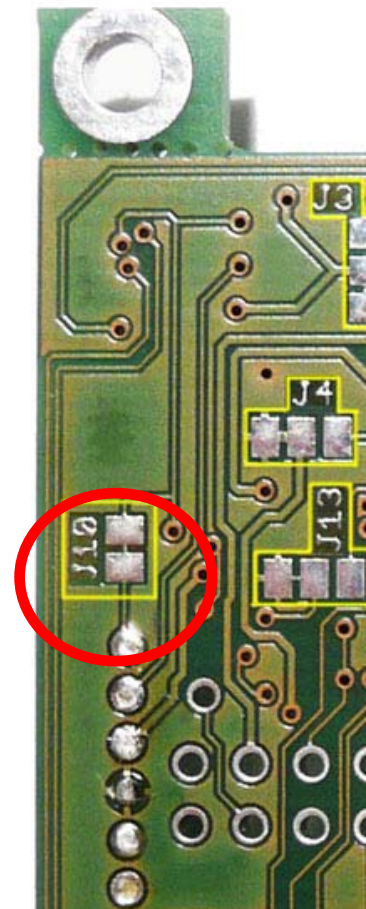
As the display lighting needs a higher voltage (at our board we create 16 volt for the lighting), some addition is needed: you may either add a manual switch or two SMD transistors and two SMD resistors. By default we ship our board with permanent lighting. If you did not order the board with a different lighting option, you need to modify/upgrade it by yourself.

By default there is a direct connection from the PowerBooster to the display with its integrated lighting. The lighting is separated from the other electronics of the display and may therefore be disconnected during run time.

As there is a direct connection by default, you first have to cut it: Just cut the connection between the two small pads of Jumper J10 showing in the circle at the photo below.

Switching the lighting with a manual switch:

Solder two wires at the pads left and right of the wire you just cut (=the pads inside the circle) and connect these two wires with a manual switch. You now can switch the lighting on and off whenever you want. Switching it off will lower the current usage of the complete module by about 40mA.



Switching the lighting with the microcontroller:

Instead the manual switch you may use your microcontroller to automate the lighting (e.g. microcontroller switches the lighting off after 10 minutes without any user input – as soon as a key is pressed the lighting will be switched on again).

As the lighting need a higher voltage, we cannot use a direct connection from the controller to the display lighting. We will need transistors as electronical switches. In detail you need:

- 1 x SMD-PNP-Transistor BC858A (size SOT 23)
- 1 x SMD-NPN-Transistor BC847A (size SOT 23)
- 2 x SMD-resistors 5,6 KOhm (size 805)

We did prepare the board for the usage of the **port B.7**. **First cut the existing bridge of J10** as shown at the last page.

Then you solder to the top side of the PCB the two transistors as shown at the following photo. Important: the upper transistor should be the **NPN-Transistor (BC847) = T1** and the lower one should be the **PNP-Transistor (BC858) = T2**.

Next you need to solder in the resistors to the PCB. Thats it.

By connecting your module to power the display shall be dark. If you now define port B.7 as an output port and if you set the port to high-level, the display lighting should become alive. Of course port B.7 cannot be used now for anything else (see also next page).

Please be aware: in our sample programs port B is being used for the display data. Until now port B.7 was unused and was not being defined as an output port. You need to change the definition of port B.7 now to Output.



Example for our Bascom code: Change the existing line

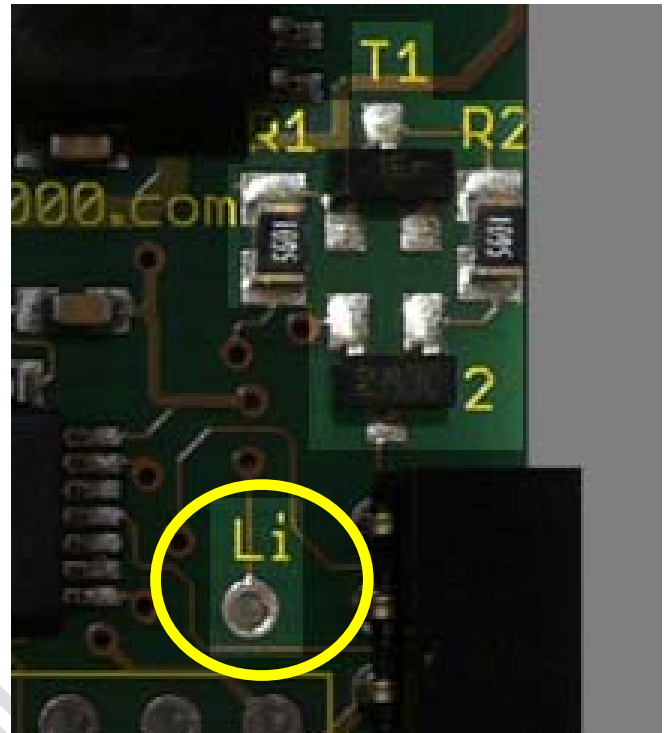
Ddrb = &B01100110 into **Ddrb = &B11100110**. With this your port B.7 is as an output port. Now, with the line **Portb.7 = 1** or **Portb.7=0** you may switch it on or off.

Changing the port for driving the display lighting:

When you need Port B.7 for something else but you would have another port left which you would like to use to drive the display lighting you may change this as we prepared the board for this.

Located at the top of the board (near the frame) there is a soldering pad marked with “Li” (see yellow circle in the photo). When you cut the wire from B7 to the “Li”-Pad with a knife (wire is located at the opposite side of PCB - display side) then you may set your own connection from any port you want to the pad “Li”.

Another interesting alternative is the driving of the display lighting by pulsed signals. With this you are able to simulate a dimmed lighting. More on this follows in the next chapter.



Running (dimming) the display lighting by PWM (pulse wide modulation)

PWM is often being used for controlling the energy usage of a technical system.

First some (simplified) theory:

If you switch on and off a LED five times a second for 0.1 seconds each you will of course recognize some bothering flickering. But also, the LED was switched off half of the time and will have emitted (and used) only 50% of the energy it might have used if it were switched on all the time.

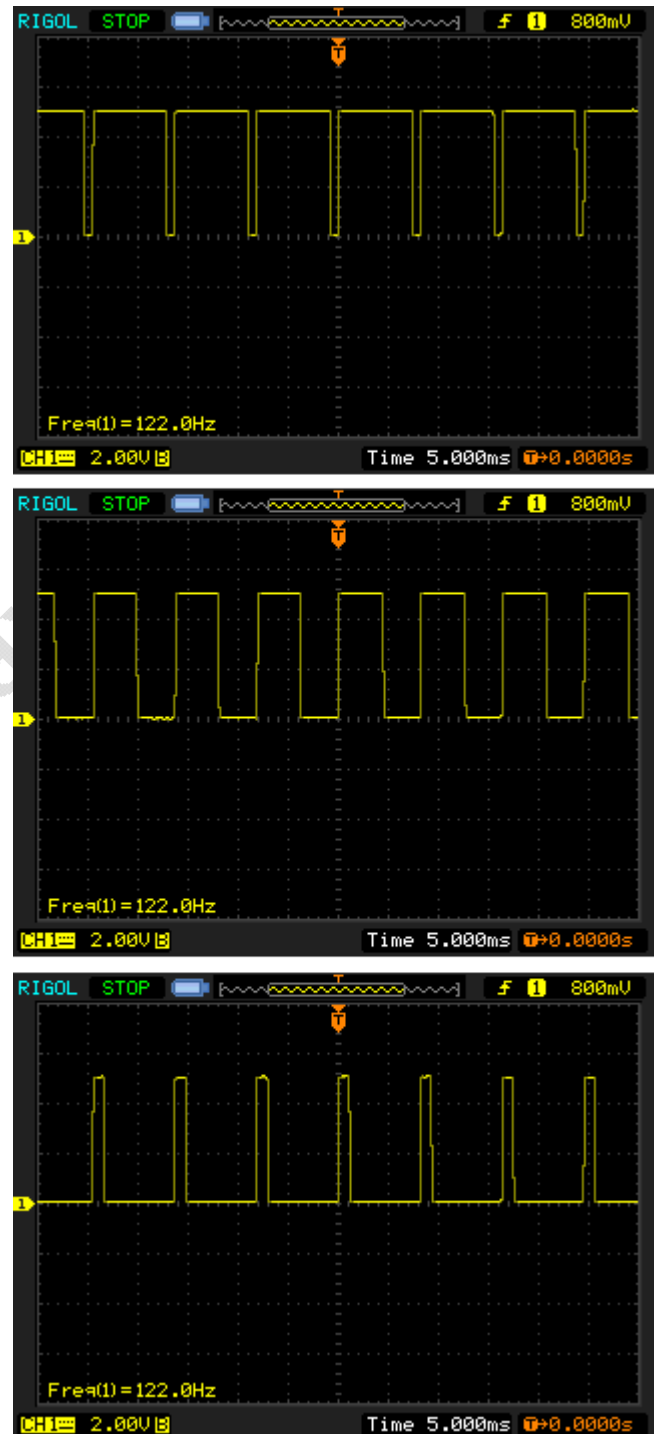
Assumed, you speed up the on/off switching time to 0.001 seconds each, you would not be able to recognize any flickering anymore – but your eye would register a LED shining with reduced power. In fact the LED is still working only 50% of its time (500 times per second switched on for a 1/1000 of a second and the same time frame switched off) and therefore it is emitting only 50% of the energy.

If you now change the variation from the 1:1 example above to a 1:3 (means: the LED is still being switched on 500x a second but for 0.0005 seconds and being switched off for 0.0015 seconds) the brightness would be reduced again. Vice versa, if the LED is being switched on for a longer time than being switched off, the brightness would raise.

This is being called PWM – the frequency does not change but the variation between high and low.

The photos of our oscilloscope beside show this in detail: In the middle there is 1:1 variation (=50% brightness). The top picture shows a 10:1 variation thus shining pretty bright and the bottom picture shows a 1:7 variation where the LED is disabled most of the time and almost very dark.

By varying the enabling to disabling time in a defined time frame the used power of a device can be controlled. In this case, the ATmega controller is also able to do this to be able to varying the brightness of the display lighting.



But wait – do not go and start writing a program which will do the needed pulsing. There is no need to do this as the ATmega controller offers hardware PWM. This is pretty cool, as you do not need to give any extra resources to the controller. It will just do the work without eating up any of your program space or time. You just need to enable it – then it will run by itself.

An example in Bascom shows this:

The following lines will enable PWM and will then dim the display from 0 to 100%, wait 500ms with full power and then dim the display from 100% to 50%. Then the program stops. Question: What will happen with the display lighting now? Will it be shut off, stays at 50% or will it be 100%. The answer will be given on the next page.

```
Gosub Lcd_cls
Call Lcd_print( "Display3000" , 1 , 1 , 2 , white , darkred)

Config Timer1 = Pwm , Pwm = 8 , Compare C Pwm = Clear Up , Compare C Pwm =
Clear Down , Prescale = 256

For I = 0 To 255 Step 5
    Pwm1c = I
    Waitms 10
Next I

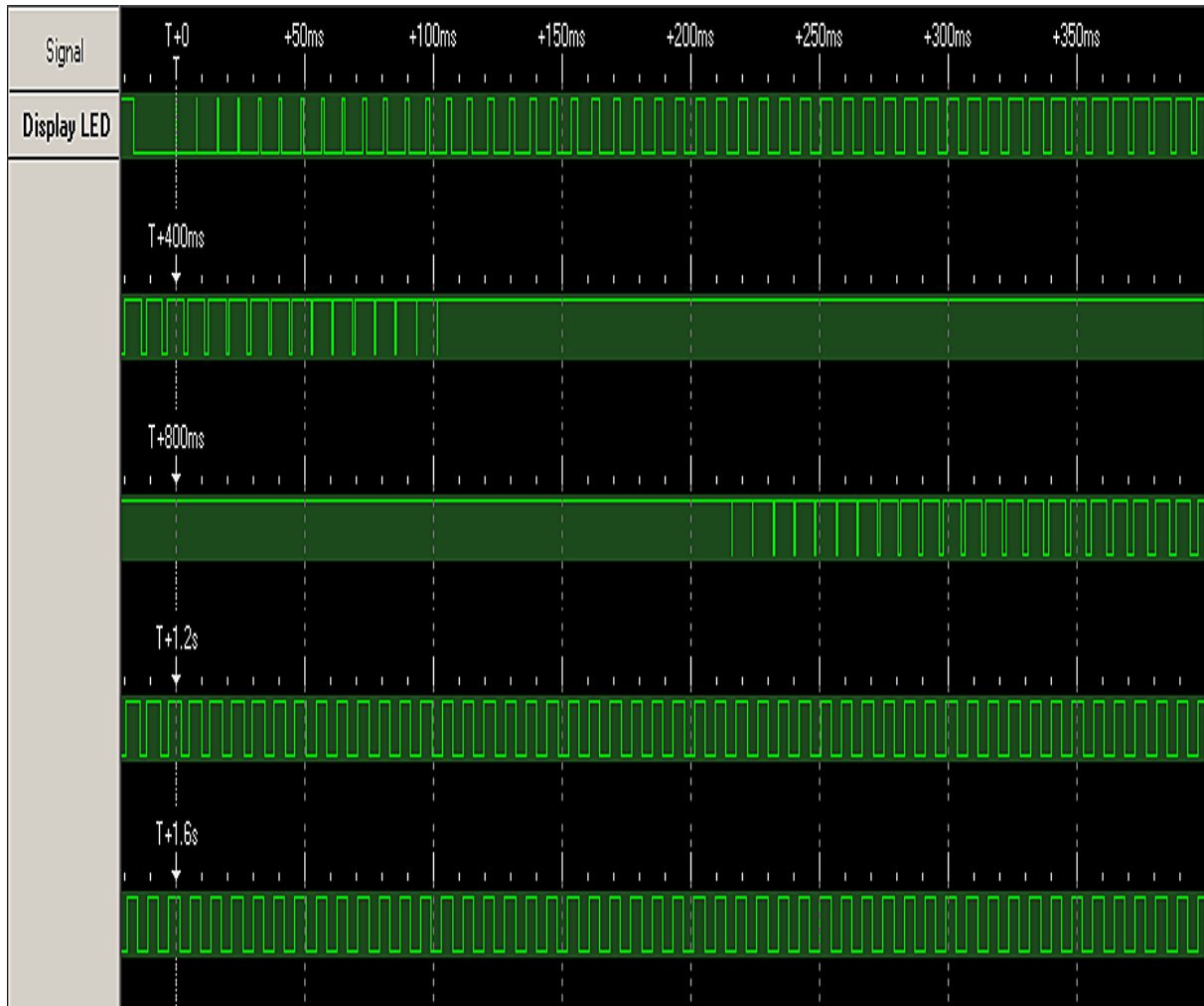
Waitms 500

For I = 255 To 130 Step -5
    Pwm1c = I
    Waitms 10
Next I

End
```

A short explanation: Port B.7 is connected to the hardware PWM channel C. With *Pwm1c* the corresponding register will be set.

To make it clearer we did record the PWM signal of the above program with our logic analyzer. The complete record is 2 seconds long, each line divided in 400ms. It is very easy to recognize, how the variation from high to low changed during the time.



The first 510ms are needed to bring the brightness up from 0 to 100% - 51 steps with 10 ms waiting time each. Then a pause of 500ms (Waitms 500) is visible, during this pause the display lighting is still at 100%. Then, the lighting is being driven from 100% to 50% in 25 steps, and then (at position 1.25 seconds) the program is finished (command END). This gives you the answer of the question on the last page: Even if the program is finished and the controller is not running a program anymore, the hardware PWM is still working with the given parameters thus not needing any resources.

I²C / TWI – Two wire interface

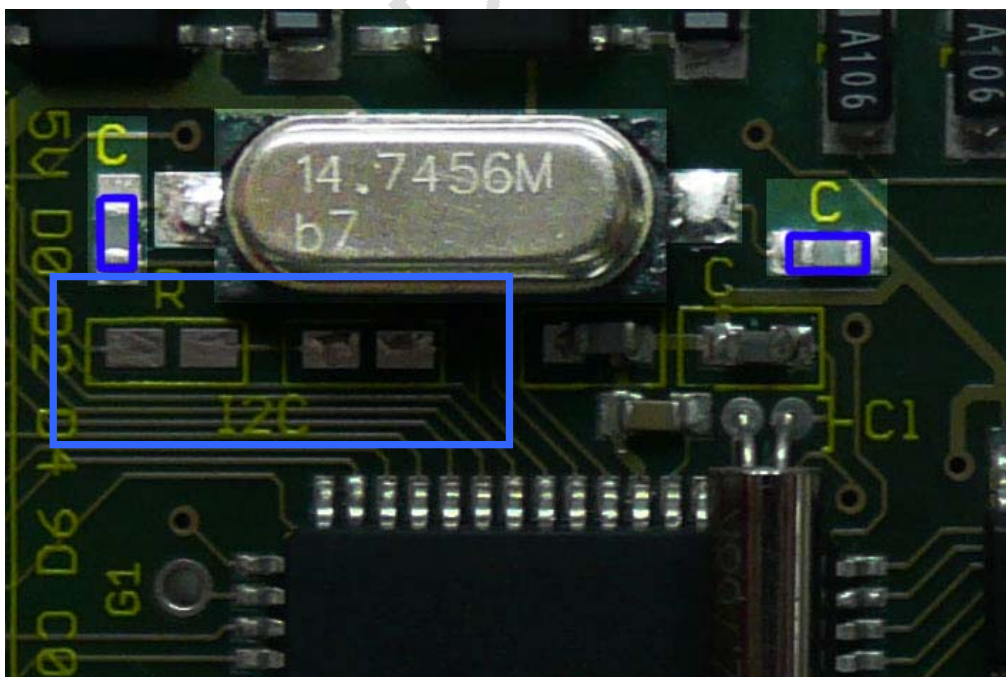
The ATmega offers a TWI, which means two wire interface. A well known TWI is I²C, (pronounced: I square C) which is compatible to the ATmega (or better: the ATmega is compatible to I²C).

This interface is called two wire interface as this bus only needs two wires for a bidirectional communication (plus GND and VCC). It is a serial synchronous two wire bus, one wire with the clock signal, the other with the data signal.

What is it for? In many modern electronic systems, a lot of different devices need to communicate with each other. If you do not want to have lots of cables running back and forth, a bus system allows to have all devices connected with each other through a bus thus allowing them to talk to each other to be controlled by a master system

A big advantage of the I²C-Bus is the easy usage: No fixed clock rates are needed thus small and fast devices can be connected to the same bus, all running different chips and run by different programming languages. That should be enough for a short explanation. The internet offers tons of information on I²C.

Our module D072 offers the direct connection to a I²C-bus on Ports D0 and D1. A I²C-bus always need pull up resistors at one device to draw the voltage to a defined level. If the D072 shall work as the master system on the bus and if there are no other pullup resistors set up on the bus, these may be soldered to the board. We are offering two places for these needed I²C pullup resistors (4,7 KOhm up to 10 KOhm). The following photo are showing the two places, where each a 10 KOhm-resistor (size 603) should become soldered to. Beside some software, there is nothing else needed for I²C.



CAN bus interface

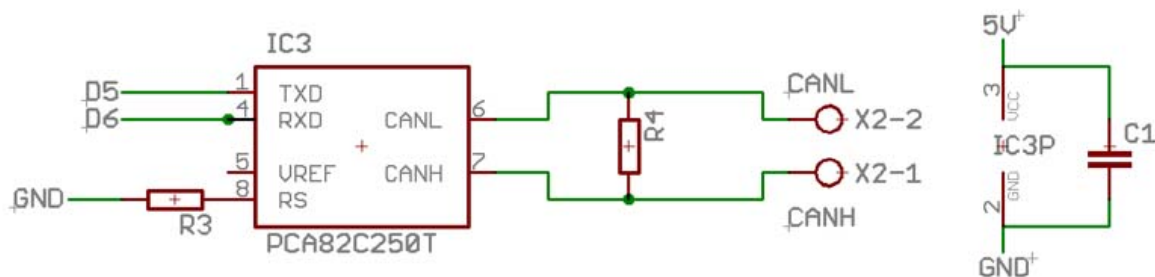
The CAN-Bus (Controller Area Network) is a so called field bus. It is a asynchron, serial bus system.

With a CAN bus you might connect almost any number of different modules. The length of the possible wires is depending on the speed of the data transfer. The maximal theoretical length is:

1 MBit/s	40 meters
500 KBit/s	100 meters
125 KBit/s	500 meters
50 KBit/s	1000 meters

The CAN Bus can only be used with the AT90CAN128 microcontroller (similar to AT-Mega128 but including a CAN bus controller) together with an external CAN transceiver.

If you ordered the board with an AT90CAN128 we will automatically solder the CAN transceiver in. This transceiver allows you to connect the system to the CAN bus. We are using the high speed controller (as ISO 11898-2), the NXP (formerly Philips) PCA82C250 which is the usually used transceiver type (=standard type).

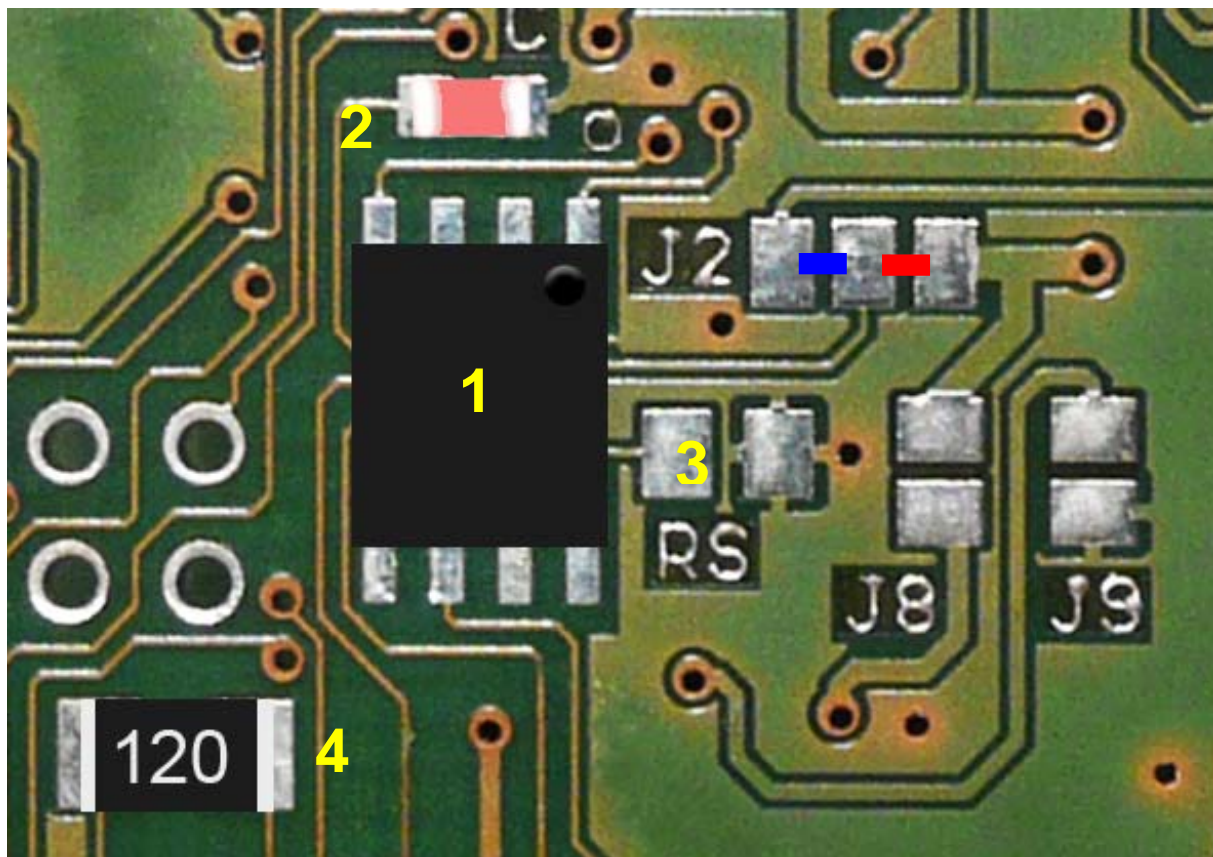


Information to resistor R4: This is the bus terminator and should be 124 Ohm – usually 120 Ohms is used here and so did we too.

R3 (resistor RS) NEEDS to be selected and soldered in by you. **You NEED to solder in a resistor here. We did not** as the value is depending on the needed bus speed. Please consult the data sheet and the *Application Note* of the PCA82C250 Transceivers which we placed on our CD.

First hints:

Highspeed:	0 Ohm – so you need to bridge (just a drop of solder)
1 MBit/s	1 KOhm
100kBit/s:	47 KOhm



Legend:

- (1) = CAN Transceiver
- (2) = capacitor 100nF
- (3) = RS resistor (must be chosen and soldered in by you – see last page)
- (4) = 120 Ohm Bus-termination resistor

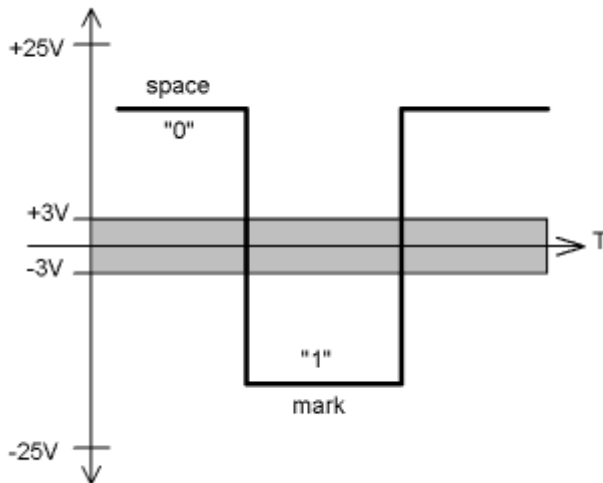
Remark: D5 and D6 are connected directly to the transceiver. As D5 and D6 are also used by two the switches, you shall reconfigure the switches (see page 12)

With solder bridge J2 you can select if the CAN Transceiver shall be connected with Vcc all the time (**red**) or –if the switch off option has been ordered – shall be disconnected from the power when the switch off is initiated (**blue**). Please check, if the Jumper J2 is closed by default – if not, you need to do this first. If the jumper J2 is not closed in either direction, the CAN transceiver will not be connected to Vcc.

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.

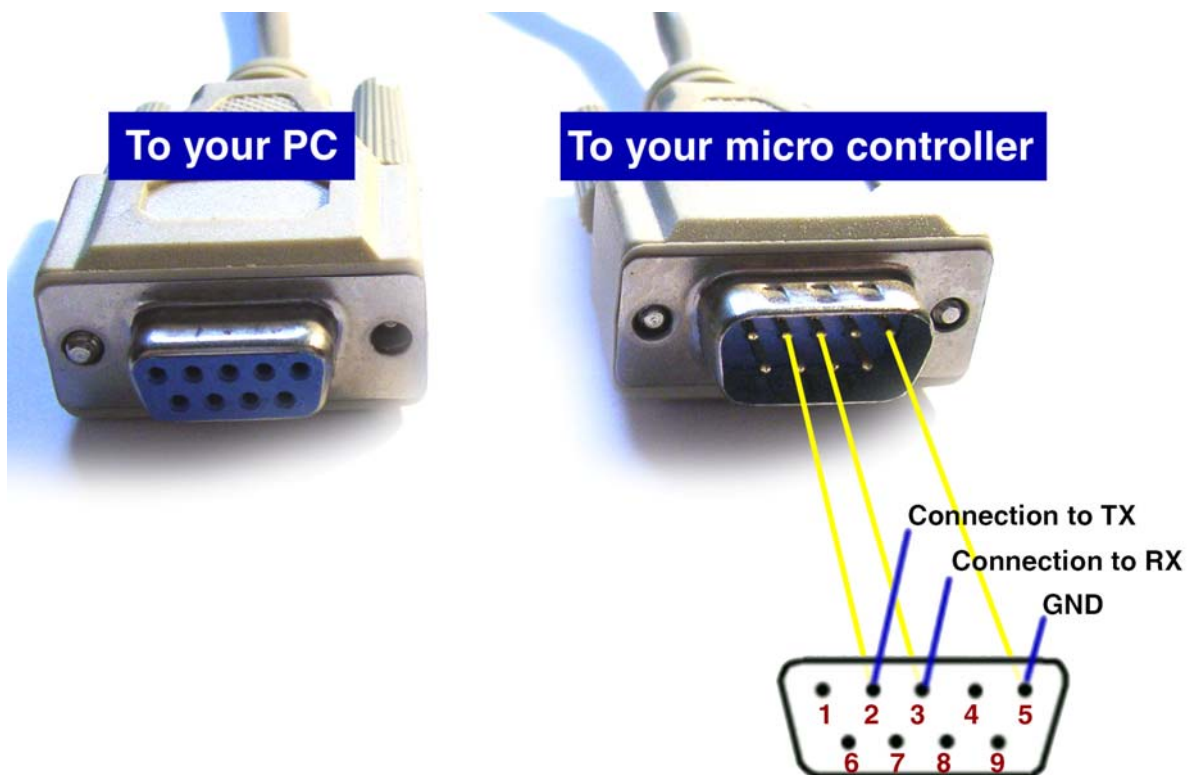
RS-232 is an asynchronous protocol, meaning that no separate clock is transmitted with the data. Both sides must know the communication speed (we use the term baud-rate) beforehand.

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 your board. The following picture will help you to build an adapter and to connect the cable to your board.



Pin 2 is the receive channel of the PC – but for this 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 128 offers two separate RS232 interfaces, we are only using interface 1 (the other one is interface 0 – see ATmega128 data sheet for this). At the ports D2 and D3 the used RS232 interface 1 of the ATmega is located. These two ports (D.2 and D.3) 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.

Caution:

The Rx and Tx pads are located beneath the pads for Port F. The Tx and Rx lines may show a voltage of much more than 5 Volt. If you accidentally connect these lines to any other pad of the board, you may damage the whole micro controller and/or the display – at least some ports will be destroyed after that. So be careful that you never use the Tx and Rx line for anything else than connecting a RS232 cable.

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[®] you may use either the Hyperterminal[®] which comes with MS Windows[®] or with Bascom[®] 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 = "m128def.dat"
$crystal = 8000000
$baud1 = 9600

Open "COM2:" For Binary As #1
Do
  Print #1 , "Hello world"
  Wait 1
Loop
Close #1
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:

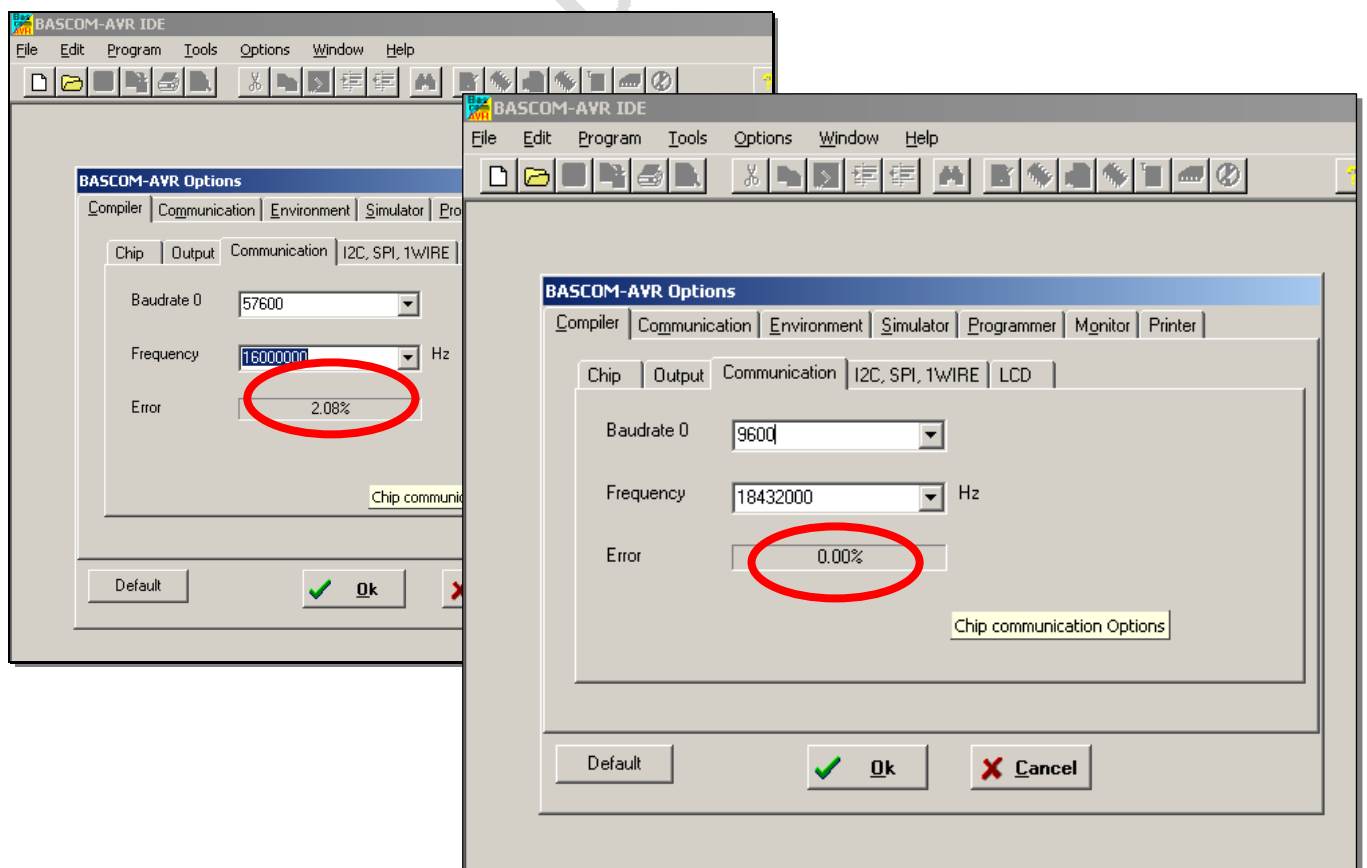
- a) The internal resonator is not very accurate and varies with different temperatures etc. So if the board runs with internal 8 MHz, using of the RS232 interface may result in transmission problems. You better use an external crystal then.
- b) Using the usual 16 MHz crystal as the external clock will result in a wrong frequency of the RS232 interface – varying by the selected baud rate. You better use 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 compiler, 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.

The following table shows you the error rate splitted on baud rate and clock rate of the controller. As black number means OK, a red number might result in a higher error rate during transmission.

Baud	Clock rate of the mikrocontroller in MHz										
	1,00	2,00	4,00	7,373	8,00	11,059	14,318	14,746	16,00	18,432	20,00
2400	0,2%	0,2%	0,2%	0,0%	0,2%	0,0%	0,0%	0,0%	-0,1%	0,0%	0,0%
4800	0,2%	0,2%	0,2%	0,0%	0,2%	0,0%	0,2%	0,0%	0,2%	0,0%	0,2%
9600	-7,0%	0,2%	0,2%	0,0%	0,2%	0,0%	0,2%	0,0%	0,2%	0,0%	0,2%
14400	8,5%	-3,5%	2,1%	0,0%	-0,8%	0,0%	0,2%	0,0%	0,6%	0,0%	-0,2%
19200	8,5%	-7,0%	0,2%	0,0%	0,2%	0,0%	-0,8%	0,0%	0,2%	0,0%	0,2%
28800	8,5%	8,5%	-3,5%	0,0%	2,1%	0,0%	0,2%	0,0%	-0,8%	0,0%	0,9%
38400	-18,6%	8,5%	-7,0%	0,0%	0,2%	0,0%	1,3%	0,0%	0,2%	0,0%	-1,4%
57600	8,5%	8,5%	8,5%	0,0%	-3,5%	0,0%	-2,9%	0,0%	2,1%	0,0%	-1,4%
76800	-18,6%	-18,6%	8,5%	0,0%	-7,0%	0,0%	-2,9%	0,0%	0,2%	0,0%	1,7%
115200	-45,7%	8,5%	8,5%	0,0%	8,5%	0,0%	-2,9%	0,0%	-3,5%	0,0%	-1,4%
230400	-72,9%	-45,7%	8,5%	0,0%	8,5%	0,0%	-2,9%	0,0%	8,5%	0,0%	8,5%
250000	-75,0%	-50,0%	0,0%	-7,8%	0,0%	-7,8%	10,5%	-7,8%	0,0%	-7,8%	0,0%

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



Hints for selecting the correct crystal:

Any timing is always ascertained by dividing the main clock frequency. As the internal 8 Mhz clock generator of the microcontroller is not ver accurate (just a RC-combination) you **always** should use an external crystal if you want to use RS232 or if you need an exact timing for a clock etc.

Otherwise the selection of the speed of the crystal is only dependent of the needed speed of you program or if the divergence of the error rate from the ideal # is acceptable when you use RS232.

RS232 / RS485 / CAN-Bus:

14.7456 MHz or 18.432 MHz will result 0,00% error rate and are the ideal selecting then. Unfortunately these crystals are sometimes hard to get. If you read this before you plan to buy such a module you might consider to order the board including such a crystal. This is not expensive and we then already preprogrammed the microcontroller fuses to the selected speed.

Using a 16 MHz crystal will result in a 0.16% error rate which is still OK if you plan to use 9600 baud (see preceding page) – anything below 0.5% is usually acceptable.

Timing:

An even crystal (e.g. 16 Mhz) does has another advantage: as all timings are being created by dividing, even crystals are always exact. An uneven crystal like a 14.7456 Mhz will result in minimal division variation. A clock for example will always have a slight inaccuracy then. Here you need to weigh the pros and cons. Hint: if you want to use the buildt in real time clock (RTC) of the controller, you might also consider add an extra clock crystal to the module (can also be ordered including soldered clock crystal) – then the clock is independent of the main crystal.

Needed power:

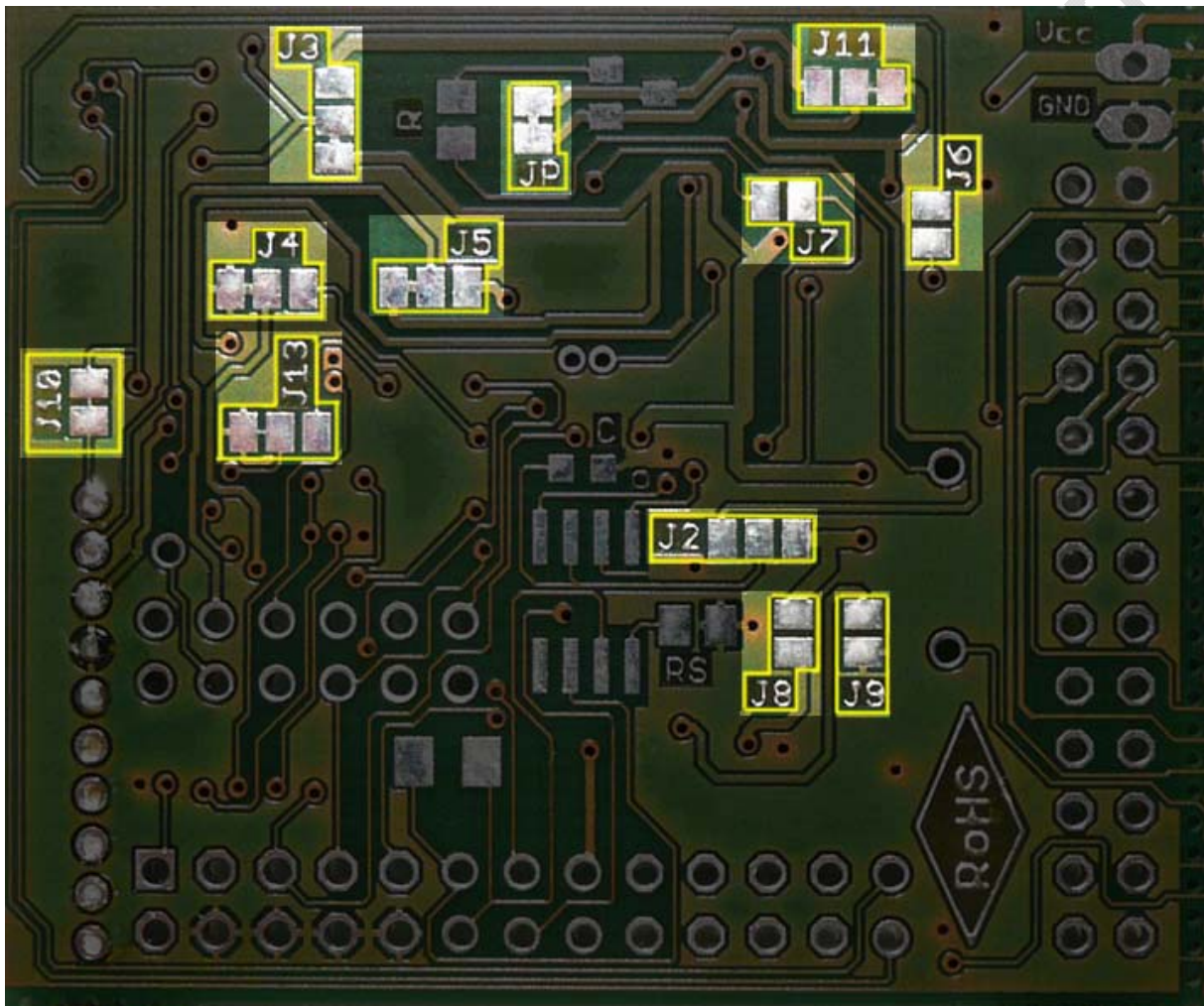
Another aspect which should be mentioned: The higher the clock frequency, the higher is the needed current of the module.

Solder bridges / jumper

Several solder bridges / jumper allow a reconfiguration due to different usage needs. Each jumper / bridge is marked with a number and is described on the following 2 pages.

The soldering bridges on the front side J1a, J1b, J1c, J1d in the switch area were already described in the chapter switches. All other jumper are located on the backside. For modifying it is needed to disconnect the display first.

Here you will find bridges / jumper J2 to J13 as well as JP.



Bridge CAN (J2):

If you ordered a AT90CAN128 microcontroller, you select here if the CAN transceiver shall be disconnected from Vcc by the switch off option (see page 13).

Bridge B6/GND (J3):

By default the RS232 Transceiver and DCDC Booster for the display lighting will be connected to Vcc 5V. By reconfiguration of this jumper field you select if the switch off option is available for this device (see page **Fehler! Textmarke nicht definiert.**). For this you need to cut the wired from the middle to the lower field and then create a new connection with a bit of solder from the middle to the upper field.

Bridge B6/GND (J4):

As B6 is also used for the display (DC-Data/Command), any other usage of B6 is very unlikely. Therefore we did not connect it to the Port B connector but connected GND instead to a pad. However you might change this configuration. If you need B6 and not GND on the inner connector field of the PCB: For this you need to cut the wired from the left to the middle and then create a new connection with a bit of solder from the middle to the right field.

Bridge B4/Vcc 5Volt (J5):

As B4 is also used for the display (Reset), any other usage of B4 is very unlikely. Therefore we did not connect it to the Port B connector but connected 5V instead to a pad. However you might change this configuration. If you need B4 and not 5V on the inner connector field of the PCB: For this you need to cut the wired from the left to the middle and then create a new connection with a bit of solder from the middle to the right field.

Bridge D3 (RS232) (J6):

Bridge D2 (RS232) (J7):

As the RS232-Transceiver is connected to the ports D2 and D3 (RxI and TxI of the microcontroller), the transceiver would affect these ports, even without a RS232 connection. If you want to use a “real” RS232-connection (high level, using of the pads RX and TX on our PCB) you cannot use port D.2 and D.3 at the same time.

For this reason, we disconnected transceiver from D.2 and D.3 by these two bridges. **This means, you need to close J6 and J7 before you can use RS232.**

D.2 and D.3 are available always at the pin row, just the connection to the RS232 transceiver is being affected by these bridges.

If you neither need RS232, nor D.2 and D.3, just let everything unchanged. You just need to remember the above if you later want to use RS232.

Bridge ARef bridge (J8):

Some (eight) ports of the controller are able to convert analog signals into digital values. If you need very exact conversions you will find at the ATmega a special input for a reference voltage: ARef. Often an external ARef is not needed (ARef then is connected with the Vcc of 5V). For this case we prepared the board:

By closing the bridge (J8) you connect ARef to Vcc (5V). If you need to offer your own reference voltage you need to open this bridge.

For the experts: Please do not use the internal interne reference voltage of the controller as long as you have connected ARef with any voltage.

Aref comes with its own 100nF capacitor.

Bridge AGND bridge (J9):

The same as with Vcc / ARef is valid for GND / AGND. By closing the bridge (J9) you will connect AGND to GND.

Bridge lighting (J10):

As long as this bridge is closed, the display lighting will have permanent lighting. By opening the bridge and soldering in two transistors and two resistors, you can switch or dim the lighting with the microcontroller. More on this at page 16.

Bridge 3V regulator with display and display electronics (J11):

As long as the middle and right field are connected, the 3 Volt regulator is connected to 5V permanently. By opening this connection and reconnecting the middle with the left field you may switch the regulator on and off by using the switch off option for saving energy (see page 13)

Bridge (J12): not available

Bridge activate tristate of display electronics (J13):

Der Pegelwandler für das Display ist permanent auf die SPI Schnittstelle aufgeschaltet. Es mag Situationen geben, wo andere Busteilnehmer das Display u.U. irritieren können. Dann lassen sich mittels Port E6 die Tristateausgänge des Pegelwandlers aktivieren und somit das Display komplett vom SPI Bus wegschalten. Hierzu muss die Brücke umkonfiguriert werden (Auf-trennen Links-Mitte und Schließen von Mitte-Rechts). Mit E6 auf Low schaltet sich das Display auf den SPI Bus auf, mit E6 auf High wird das Display vom SPI Bus abgeklemmt.

Bridge Power Off option (JP):

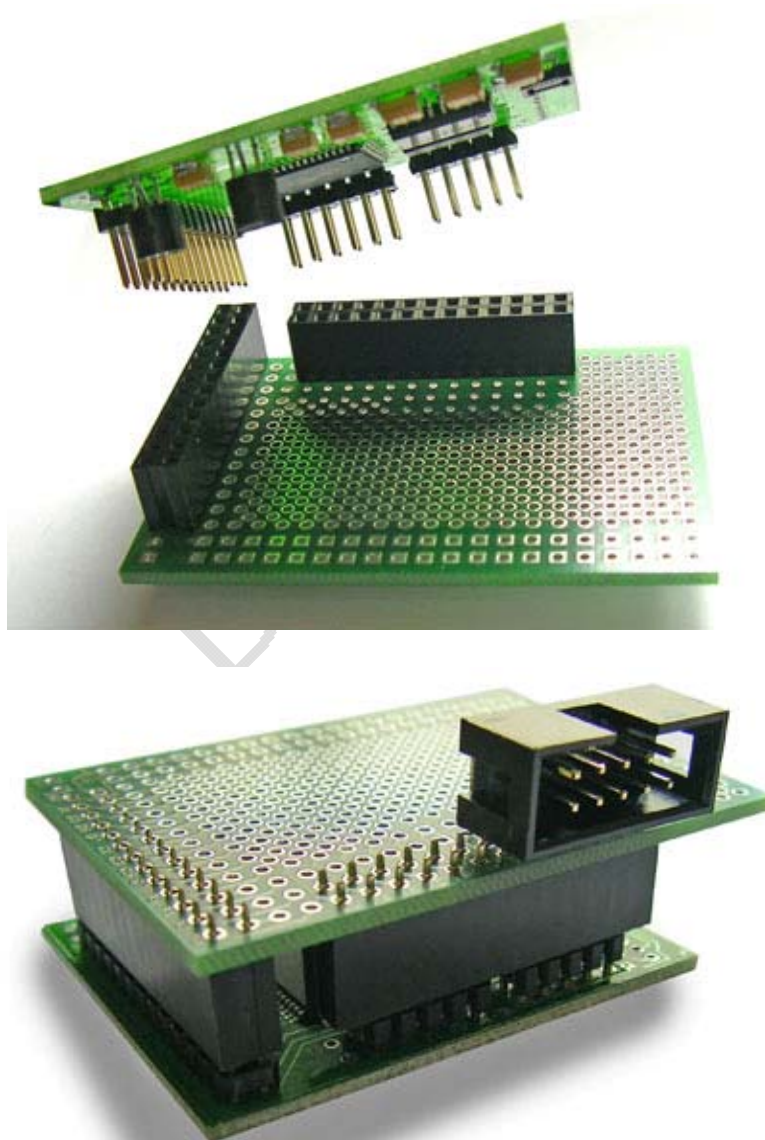
As long as this bridge is closed, all devices on the board are permanently connected to Vcc. By using the optional Power Off mode (see page 13) you can, by opening the bridge, control (switch on / off) devices by the microcontroller for saving energy.

Using additional hardware

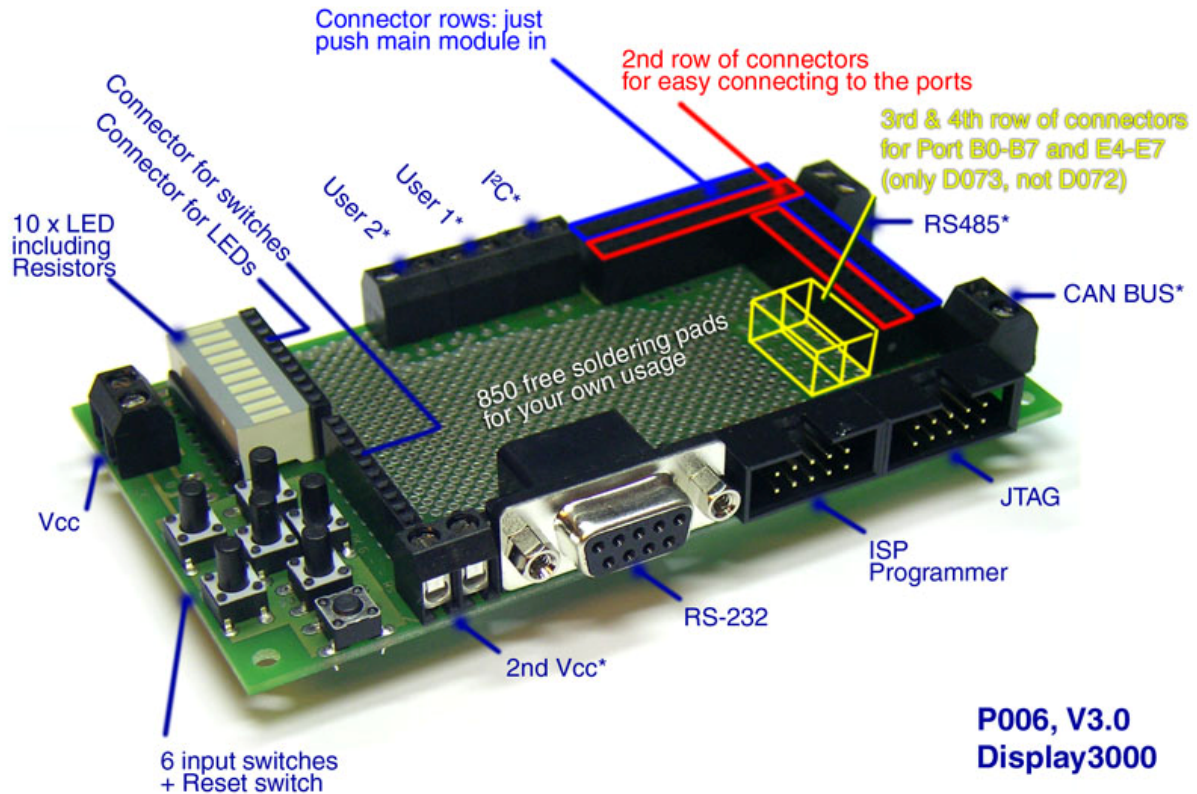
There are two additional hardware items available for the board D072 and D072:

First there is the **mini-development PCB #P005**:

This double sided PCB has a size of just 60 x 51mm and fits behind the display module. Due to the connectors which come with it you may easily mount the display module D072 to it. By using this development PCB you can easily enhance our board by some devices your application needs (like relais, transistors etc.). You may also add a standard ISP connector by just soldering it to the prepared space.



Then there is the **larger development PCB P006** which contains also 5 switches plus reset switch, a standard RS232 connector, a power input plus space for a larger voltage regulator and lots of soldering pads which allows you to set up any application you want.



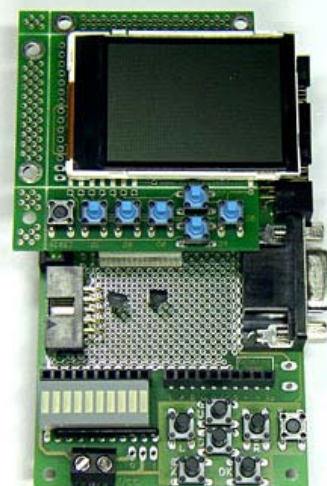
Here also, just push the D072 module to the connectors and you are able to use RS232, ISP adapter, switches, LEDs, etc. Of course you may use this PCB not only for development but also for a final application.



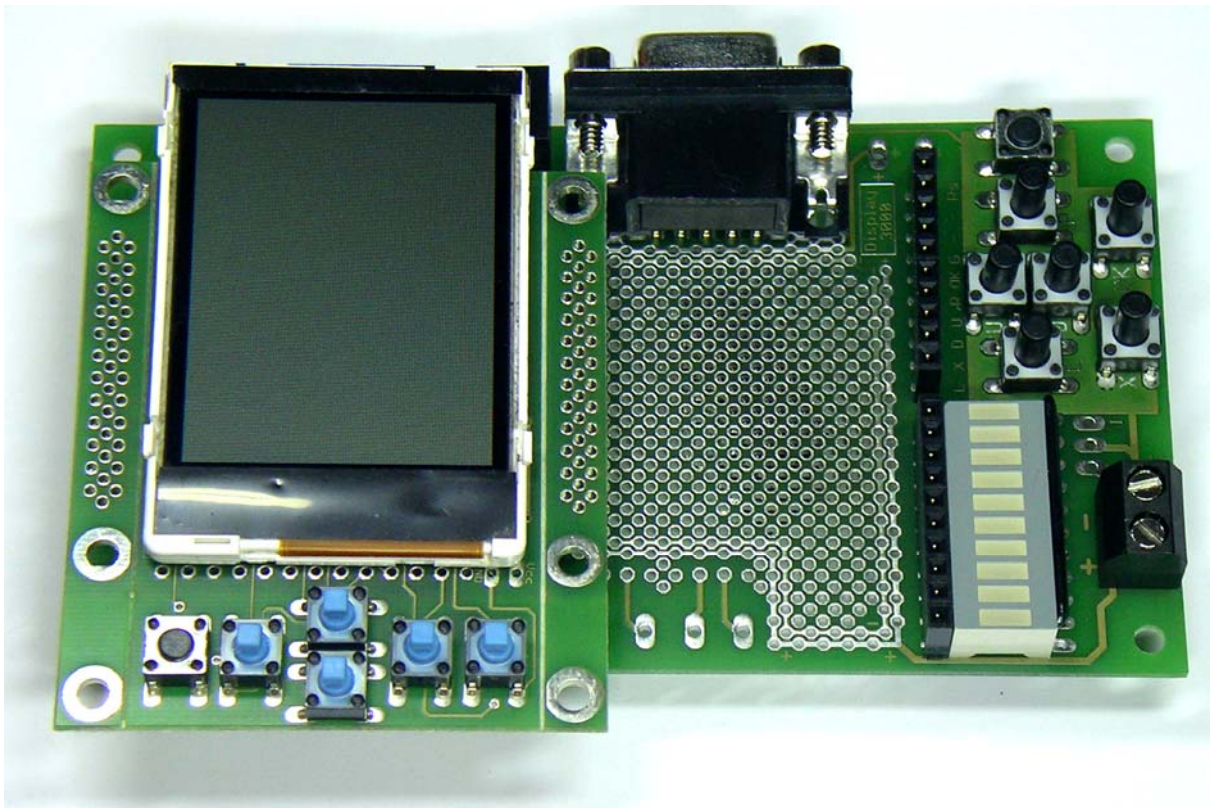
1. Place your devices



**2. Add the needed wires
either at top and/or bottom
side of the PCB**






**3. Push main modul
into place**
Changes to schematics?
Just pull out the main
module - it stays like new



And this it how it looks if a module D072 is pushed to the development board P006.

Option: ISP Programming interface

An ISP programming interface is needed to transmit the compiled program from your PC to the module. We are offering three different ISP variants:

Parallel port (printer – Centronics)	Serial programmer for RS232	USB-Interface (consists of USB-RS232 adapter together with our serial programmer)
		

The most flexible (and least expensive) programmer is the parallel programmer. Unfortunately this interface is not build into newer computers anymore. Then you should consider of purchasing a serial or USB programmer.

The removal of the mounting frame

The mounting pads has been prepared by perforating them and can easily be removed by just breaking it off with a plier. Remove the display before you do this to avoid any damage to it.



The above photo shows you two alternatives: left: as you get it delivered, middle: without mounting pads, right: no mounting pads and removed switch area.

Hint for a cleaner and smoother removal: Before removing the pads or the switch area: use a sharp knife to carve the surface of the PCB once - just follow the perforation.

Switch area:

The same applies to the switch area. If you do not need them, just break it away. Before you do this you should use a knife to cut the 8 wires of the switches at the perforation area – or even better: Cut the PCB from both sides with a knife once as explained in the paragraph above.

The current need of the module

The display lighting of course needs power to be able to produce light. The total current needed is depending on several factors:

- a) **Speed** of the ATmega Controllers (the faster the higher the current need)
- b) **Display lighting:** The display lighting of course also needs power to be able to work. The lighting at our module is being driven with 16 Volt. We produce this by a 2 x voltage doubling. Well, there is no perpetuum mobile – so the needed current will raise by 2x too. With a less bright lighting, the needed current decreases.
- c) Switch off of display and RS232. You may order the switch off option for deconnecting the display and other devices from Vcc if not needed. If you then send your microcontroller into sleep mode you can bring down the current usage to a few μA which will also allow battery driven devices.

The following table shows some variable parameters with its consequences to the current need:

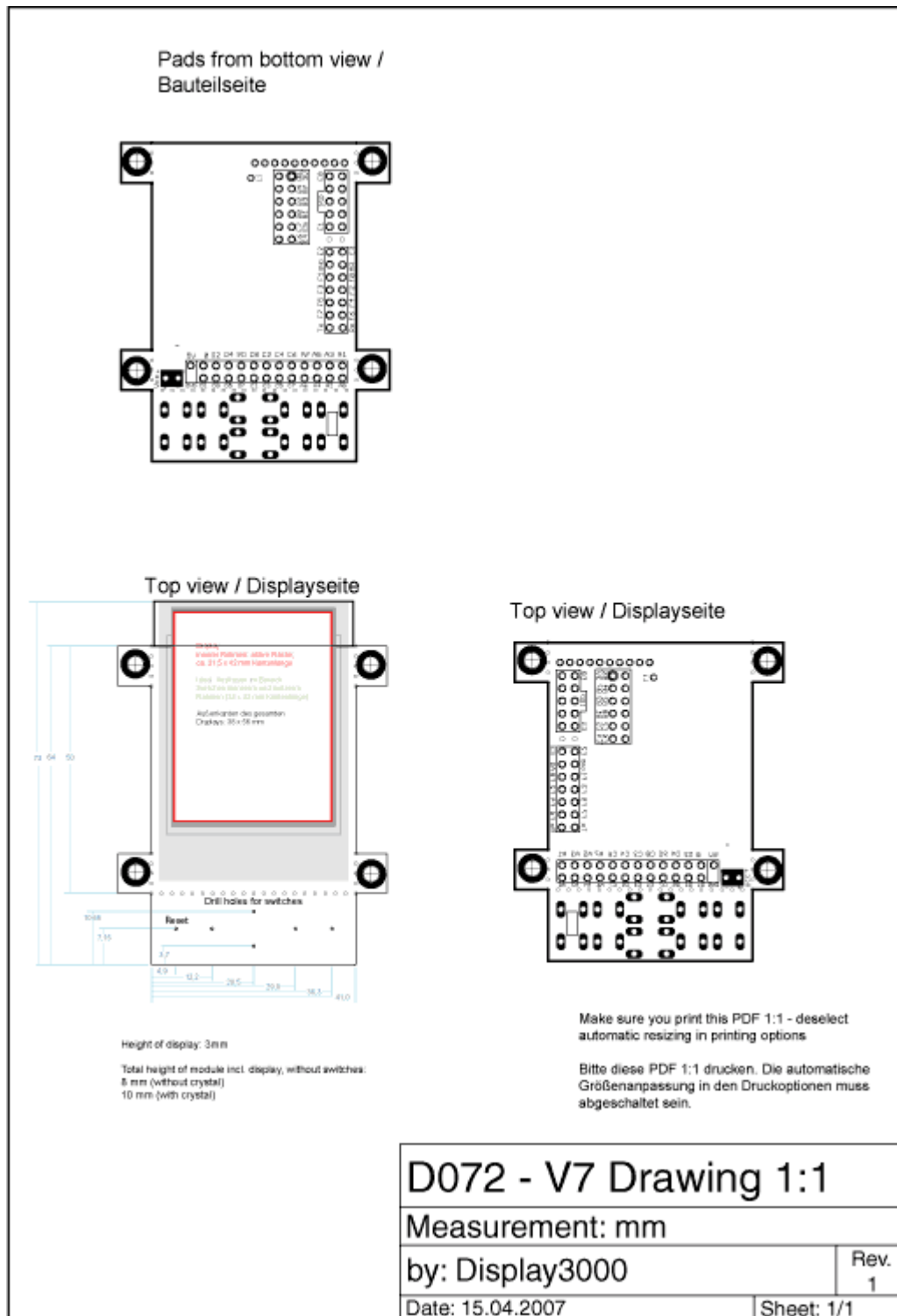
Current need for complete module (Controller 100% active, no idle-mode etc.)

Clock rate	Display lighting 0% (off)	Display lighting 50%	Display lighting 100%
8 Mhz	19 mA	41 mA	58 mA
16 Mhz	28 mA	50 mA	65 mA

Sleep Modus (display, lighting & RS232: off)	Full Speed (16 Mhz) (display, lighting & RS232: off)	Full Speed (16 Mhz) all on
Depending on how often the controller wakes up: from 100 μA	20 mA	65 mA

Drawing and pads

At the CD there is a separate PDF file with the following drawing showing this in 1:1 measurement. This following graphics does not have 100% correct size.



Technical data of display module kit:

Item D072:

Size:

With frame and switch area:	54 x 73 mm (2.1" x 2.9")
With frame, without switch area:	54 x 59 mm (2.1" x 2.3")
Without frame, with switch area:	41 x 73 mm (1.6" x 2.9")
Without frame, without switch area:	41 x 59 mm (1.6" x 2.3")
Height:	ca. 8 mm incl. Display (0.3")

Voltage:

4.5 to 20 Volt DC

Controller (depending on ordered item):

	Programm memory	RAM	Eeprom	Clock rate
ATMega 128	128 KByte	4 KByte	4 KByte	Max. 16 Mhz
ATMega 2561	256 KByte	8 KByte	4 KByte	Max. 16 Mhz
AT90CAN128	128 KByte	4 KByte	4 KByte	Max. 16 Mhz

Technical data for the display:

Display:

176 x 132 pixel, 65.536 colors
active diagonal size: 2,1" (53 mm)

Lighting:

3 white LED, color ranking 5,6

Dot Pitch H & V:

0,237 mm

Lifetime in use:

5 Years (23 h a day at 365 days/year)

Lifetime lighting with 15mA:

5.000 hours (on our board it runs with 12 mA
thus the life span will be approx. 10.000 hours)

Temperature Display storage:

-30°C to +80°C

Temperature Display usage:

-10°C to +55°C

Temperature Display usage (enhanced):

-20°C to -10°C and +55°C to +65°C *

* = reversible (= temporary) malfunction (e.g. dark picture)

Contrast Ratio Reflective Mode:

Min: 6; Typ: 11 (without lighting)

Contrast Ratio Transmissive Mode:

Min: 40; Typ: 70 (with lighting)

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

Author of this manual: Peter Kuesters

© Peter Kuesters

© www.Display3000.com