Supplement (hardware manual) for module

D071x

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

Version 1.0 (for Board D071x – V1)
14. September 2009

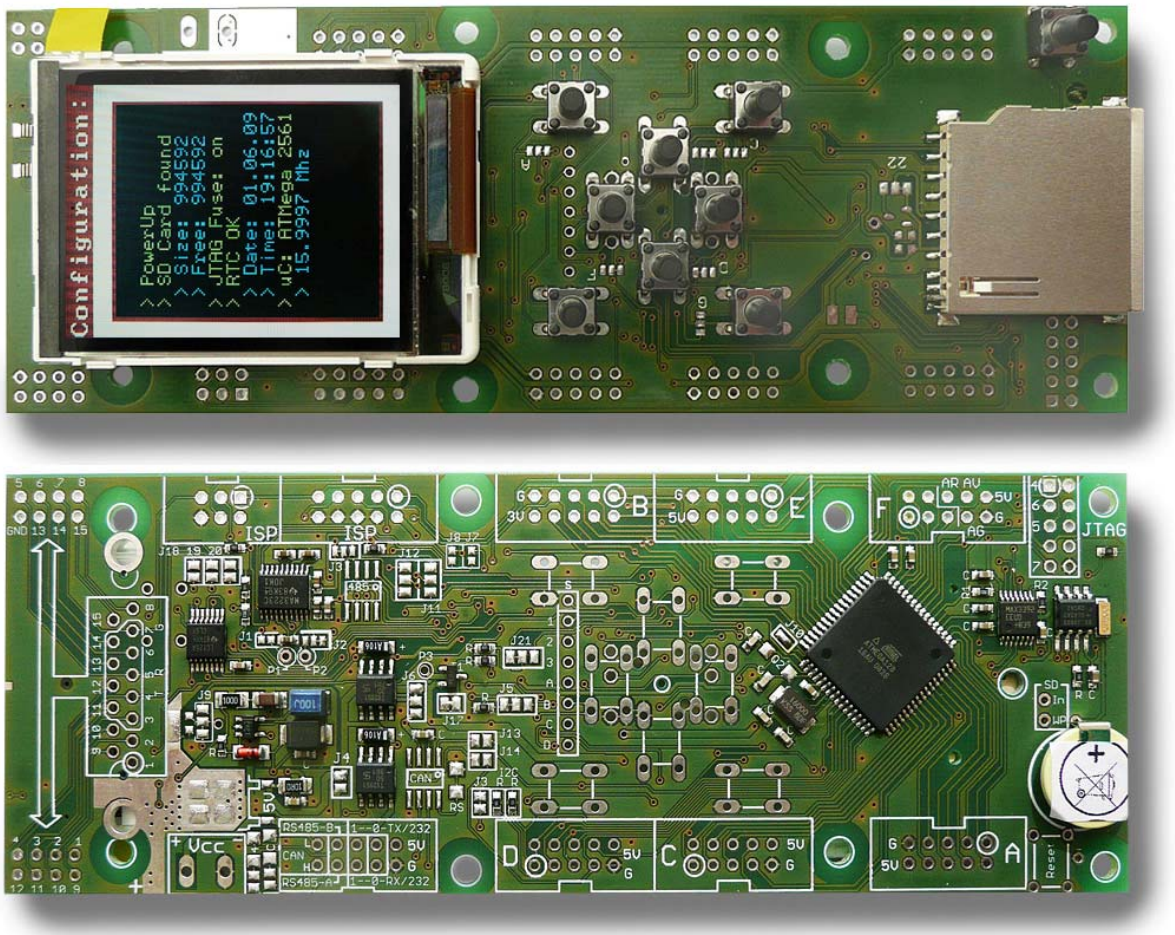


Table of content

Congratulations for buying this D071x microcontroller module including color graphics TFT.

The D071x is a multi function module where finding something equal in the world will be difficult, if not impossible. It has a practical attitude and is immediately usable for smaller or mid size series, in industrial products and in research and development.

Together with the BS600 housing and the membrane keyboard (see picture beside with a CD for a size comparison) you will be able to create a prototype which looks like a professional device manufactured in thousands. Just calculate through how much it would cost you to create a device like this, including housing and keyboard from scratch.

By the way, starting at approx. 50 PCB (any number below probably would not be economical), we can change our D071x PCB design and produce it with whatever change you need. Anything is possible. Just contact us if you are interested. Also the keypad can be either completely changed or just be delivered with a different printing (e.g. your company logo or different printing at the keys.). For many years, we have been developing and producing products for lots of customers from all possible industries.

Options:
Depending on the ordered options, your module may contain the following parts:

- Atmel microcontroller with 128 / 256 Kbytes of program memory and 4/8 Kbyte of RAM
- Transflective (=also readable in outdoor situations e.g. sunlight) full colour TFT display with 176 x 132 pixels resolution
- 2 x RS232
- 1 x RS485 halfduplex
- CAN bus
- I²C bus
- Buffered real time clock (RTC)
- Separate clock crystal for TOSC1/TOSC2 (not needed on ATMega2561/AT90CAN128 if using real time clock)
- External clock crystal for frequencies of 16 or 14.7456 Mhz
- Interface through push button switches and/or a joystick
- Elegant housing and membrane keyboard

You must complete this pre-soldered kit by mounting the display and soldering the connectors you need for your special requirements.

This manual shows you the configuration and gives a lot of tips and hints to the functionality of this board. Some parts of the boards are shown in schematic where non important parts and information are left out (this makes it easier to concentrate on the important things). The complete schematics with all used parts are shown in our complete schematics on the CD (only available for customer of this board).

As the directions left, right, upper, lower are identical with what you see, please place the PCB in front of you, so the display faces the table and the imprinting of the PCB is readable for you.

About the photos in this manual: The photos in this manual are made from two different PCB revisions. Both are 99.9% identical – there are only a few minor differences (mainly on the printing on the board). You probably will not recognize it, we are just mentioning it in the unlikely case you spot a difference between your board and the photos.

CAUTION: If anything in this manual is printed in red, you should better read it to avoid problems later and/or damages to your board.

Important hint: Please skim over the complete manual first to get an overview about what it contains and where to find what. Then go into detail with the second read.

If you are very curious, connect (after the first quick read of the manual) some voltage to the VCC connector to see the board running (it has been preprogrammed by us for testing). Before you solder in anything or reconfigurate anything, you should read the manual in total.

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

Delivery content

What do you get delivered :

- 1 x assembled PCB with micro processor ATMega microcontroller, etc...
- 1 x 2.1" TFT color display
- 9 switches
- Enough connector bars (0.1" spacing)
- 1 x ISP connector (2x5, 0.1" spacing)
- CD with sample software, documentation, utility software

Optional features

By default the board comes with a ATMega128 microcontroller. You may change this during your order to a pin compatible type:
ATMega2561 (256 Kbyte program memory, 8 Kbyte RAM, 4 Kbyte Eeprom etc.) or
AT90CAN128 (similar to ATMega128, but with an integrated CAN controller)

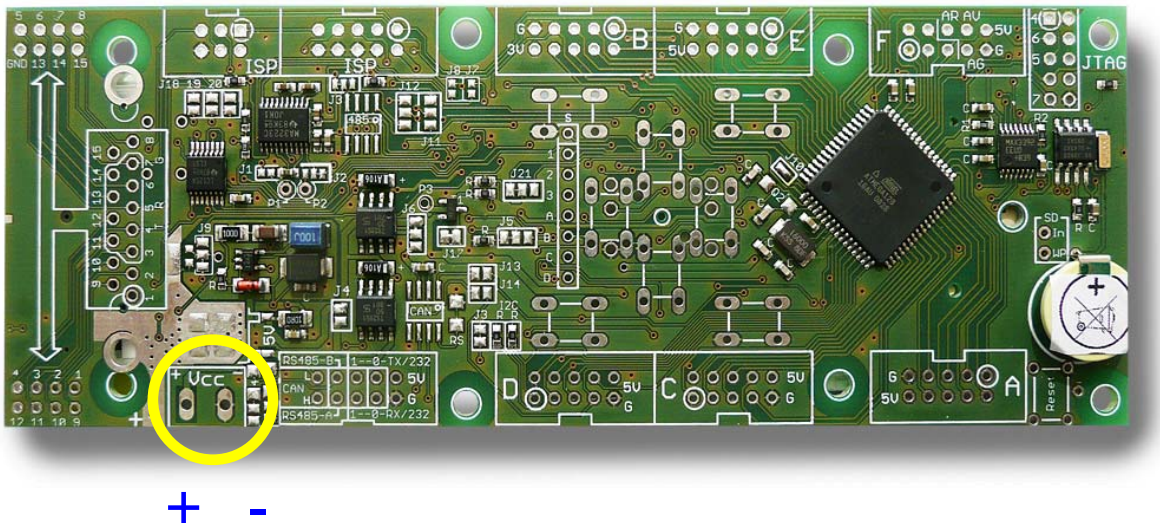
Also available as an option during the order:

- External crystal with 14.7456 MHz or 16.0 Mhz
- Clock crystal 32 KHz for internal real time counter
- Real time clock (RTC) with lithium battery as a buffer (always the correct time and date for several years, even after removing the device from the power source)
- SD card module
- RS232-transceiver
- RS485-transceiver available as a 5V or 3-5V type
- CAN-Bus-transceiver
- 5V voltage regulator for voltages >12 Volts
- Switch off option for parts of the PCB (power save)
- Joystick
- Membrane keypad
- Housing
- Programming interface
- Set of shrouded headers

Initial operation

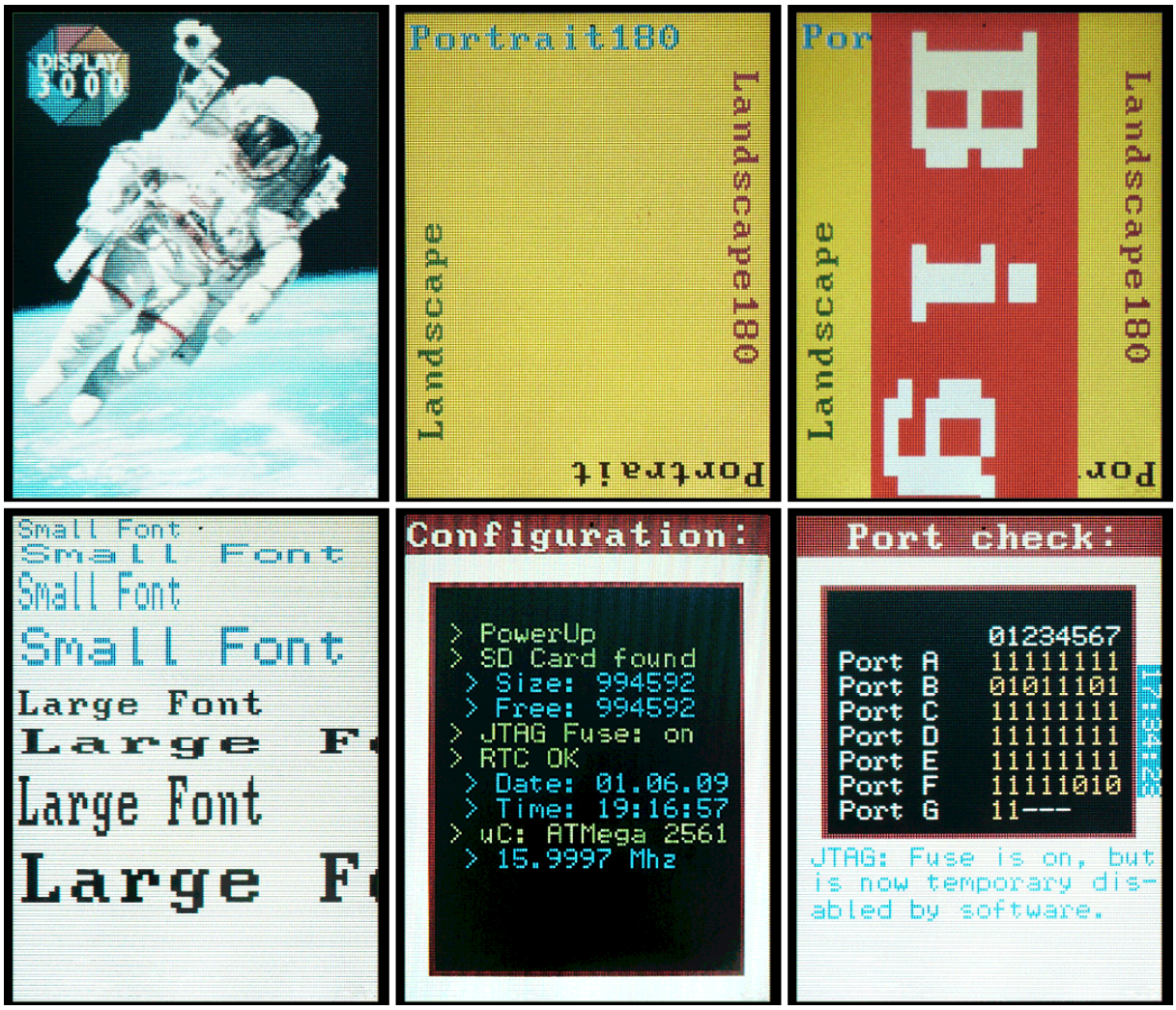
Connect any voltage between 5 and 12 Volts to the connector VCC (lower left area of PCB). The left connector is + the right connector is GND.

Comment: the PCB below might show parts you will not find on your PCB as these are optional parts which you did not order.



We already pre-programmed the module. There is a test program running on the module showing different font sizes, photos, available components etc. – see pictures on next page).

Remark: It is always difficult to make photos of TFT screens. In real the picture quality is much better than on the photos shown. The picture is more crisp and more brilliant than on our photos.



The test program shows, among other things, if a SD card is available (showing the size of the card and the free space) and if the RTC is available on the board. The shown date and time is not correct and has to be set by you.

SD card: If you place a SD card to the module it shall be recognized, the size of the card will be showed and to finish the test a file called “README.TXT” will be written to the card (you may check the TXT file by reading it from your PC). If the program shows “SD Card found” and then stops, your card has been recognized, but cannot be correctly initialized. The card is either damaged or needs to be formatted (see our hints on this on page 54)

Also the status of the JTAG fuse is shown. This is not relevant for this testing software as we are switching off JTAG by software in our program (we need a switched off JTAG for accessing the SD card). If you do not own a JTAG debugging tool you may switch off JTAG permanently with the JTAG fuse (see you ISP programming software).

The final port check shows you the status of all ports. You may, for example, push the switches into their placed (no need for soldering, they do have good contact without soldering them in) and check which port changes from 1 to 0 when you press the switch.

Quick guide: often used configurations

Because of the many different possible configurations, the manual of this module became quite large. There is some danger that you will not find the needed configuration option. Because of this, we are mentioning here, early enough, the most needed configuration and what you have to change on our board to use it. There is always a reference to the page with more detailed information beeing showed.

1) Usage of the serial RS232 interface UART1 (e.g. the SubD connector at the left border)

- Jumper J18, J19, J20 need to be closed (see page 20)
- Jumper J7 and J8 need to be closed (see page 35)
- UART0 cannot be used easily as these ports are also used by the programming interface (see RS232 chapter)

2) Usage of switch off option (for power saving)

- Open Jumper J17
- Reconfigure Jumper J6 (see page 47)
- Usage: Set PortE.3 as an output port
- PortE.0 = 1 switch off the unneeded parts (Caution: always close any open files on SD card)
- PortE.0 = 0 switch them on
- After switching on: as display and SD card were without power, they need to be re-initialized first before you may use them.

3) switch the display lighting

- reconfigure jumper J9 (see page 45)
- Switch lighting with Port B7

4) membrane keypad: usage of the red button for waking up the device

- You need to reconfigure the keypad-jumper D (see page 28). The red button then is connected to port E7 which is an interrupt port which can be used to wake up the microcontroller from a standby mode.

5) Usage of the analogue-digital converter (AD converter)

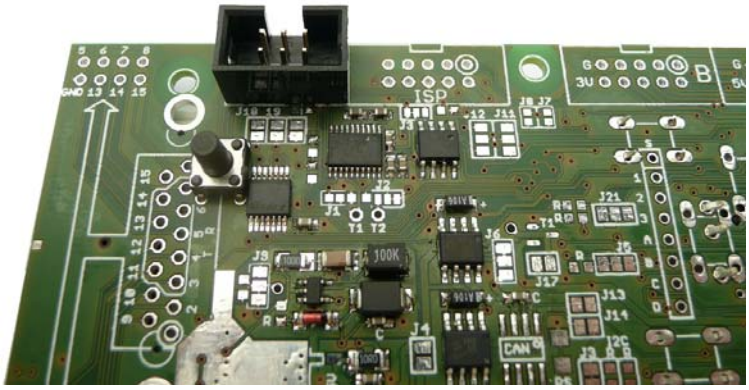
- AVcc and AGND must be connected (see page 43)
- ARef must be selected (either internal reference or connected to Vcc)
- The first four AD converters (Port F0 to F3) are immediately usable. For using the converters on ports F4 to F7, you must switch off JTAG first.

Connection of the programming interface

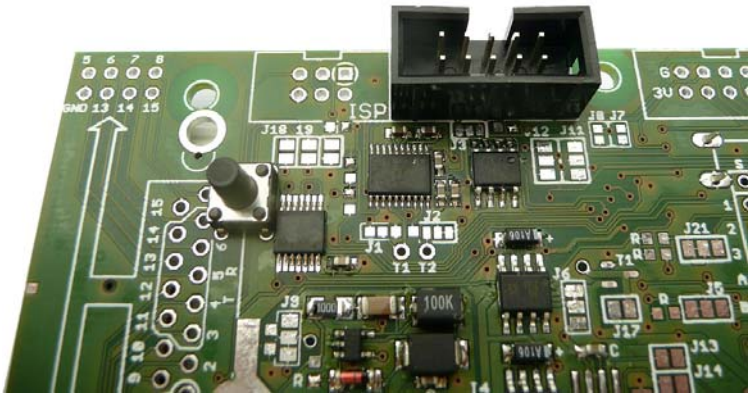
For programming the microcontroller on our PCB, you need, beside the programming software (more on this in our programmers manual), a programming interface. This programming interface is responsible to upload the software you have programmed on your PC into the microcontroller.

Usually a ISP programming interface is being used for this. This ISP interface connects your PCB to your PC.

You might either use a 6-pin or a 10-pin ISP connector. Both kinds are standardized. We prepared our board for both variants.

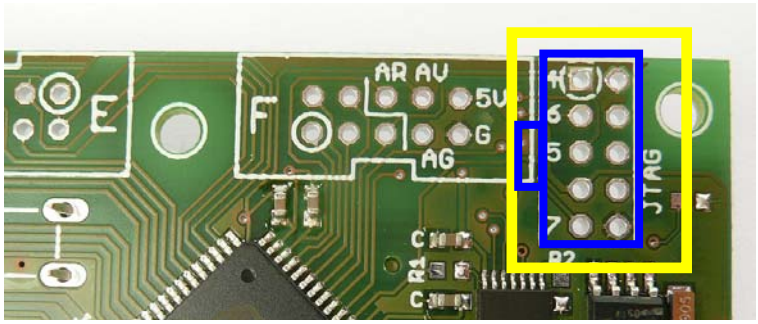


The upper photo is showing the position of the 6-pin connector; the 10-pin connector would be placed beside this one as you can see on the 2nd photo. At both shrouded headers the opening shall show towards the border of the PCB.



If you want to program with a JTAG programming interface or if you want to use the Ports F5, F6, F7 (see also the chapter analog inputs on page 43), you may access these ports as the connector in the upper right corner of the PCB beside PortF (see lower photo).

As for space problems, you will not be able to solder in a shrouded header here. You may use only 2x5 pins with 0.1" spacing.
If you solder in these pins, please make sure you are not accidently connecting these pins with the metal cover of the SD card holder on the backside of the PCB which is connected to GND.



For plugging the JTAG connector, the "nose" of the connector shall show towards the left side (see blue marking).

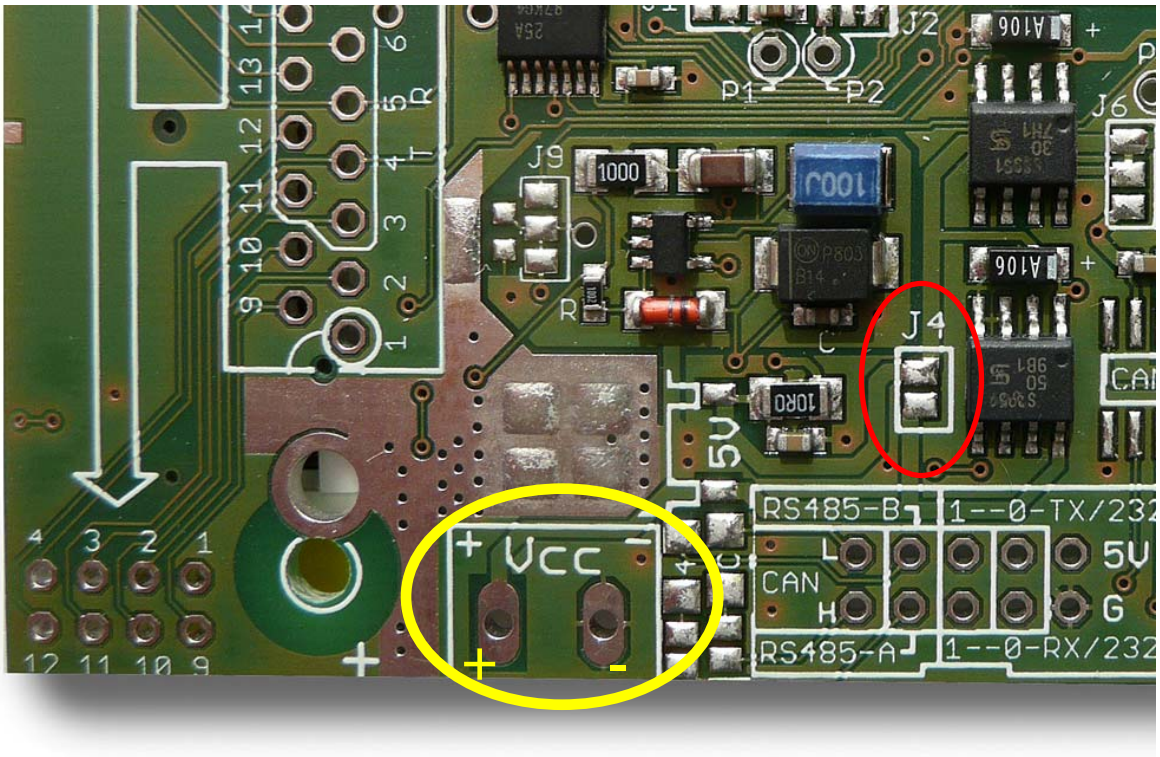
Voltage supply

The PCB is equipped with two low drop voltage regulators to allow you an easy usage and to protect the sensitive display.

You may connect 3,5 up to 18 Volt at the connector VCC-GND. For the regular operation you shall connect approx. 6 Volt up to 12 Volt. You will find more on this on the next page.

The used voltage regulator is a very low drop regulator. Thus, you may need to connect only approx 5.5 to 6 Volts to get a stable 5V output.

The default 5V regulator (see on right side beneath the jumper J4) is allowing a continous operation with up to 12 Volts. As there are no cooling options the regular is not available to emit more heat to the air. If you want to connect 12 Volts or more you shall install a larger voltage regulator (or order the board with a larger regulator). You will find more on this on page 13, at the chapter “large voltage regulator).



Running on battery

If you are providing 5 Volts or less (e.g. if you want to run the board on battery), you may bridge the 5V voltage regulator by closing jumper J4. The provided voltage will now be connected directly to the electronic parts.

Caution: If you ever connect a voltage higher than 5 Volt while jumper J4 is closed, you will damage or destroy important parts of your board.

Needed voltage

According to the datasheet, the microcontroller needs a minimum voltage of 4,5 Volts when running on 16 Mhz. With a slower speed, you may use a lower voltage. Our experience shows that the controller can runs with approx. 3.5 Volts and 16 Mhz without a problem – but remember: this is beyond the specs and nobody guarantees a save and flawless running. For critical applications, you should always stay in between the specs.

The other parts on the board do need the following voltage:

- Display: 3 Volts (3 Volts regulator is present on the board)
- SD card: (3 Volts regulator is present on the board)
- RS232: 3,3 to 5 Volts
- RS485: depending on the ordered transceiver: either 5 Volts or 3 to 5 Volts
- CAN: 5 Volts
- Real time clock RTC: 3 to 5 Volts
- Booster for the display lighting: 3 to 5 Volts

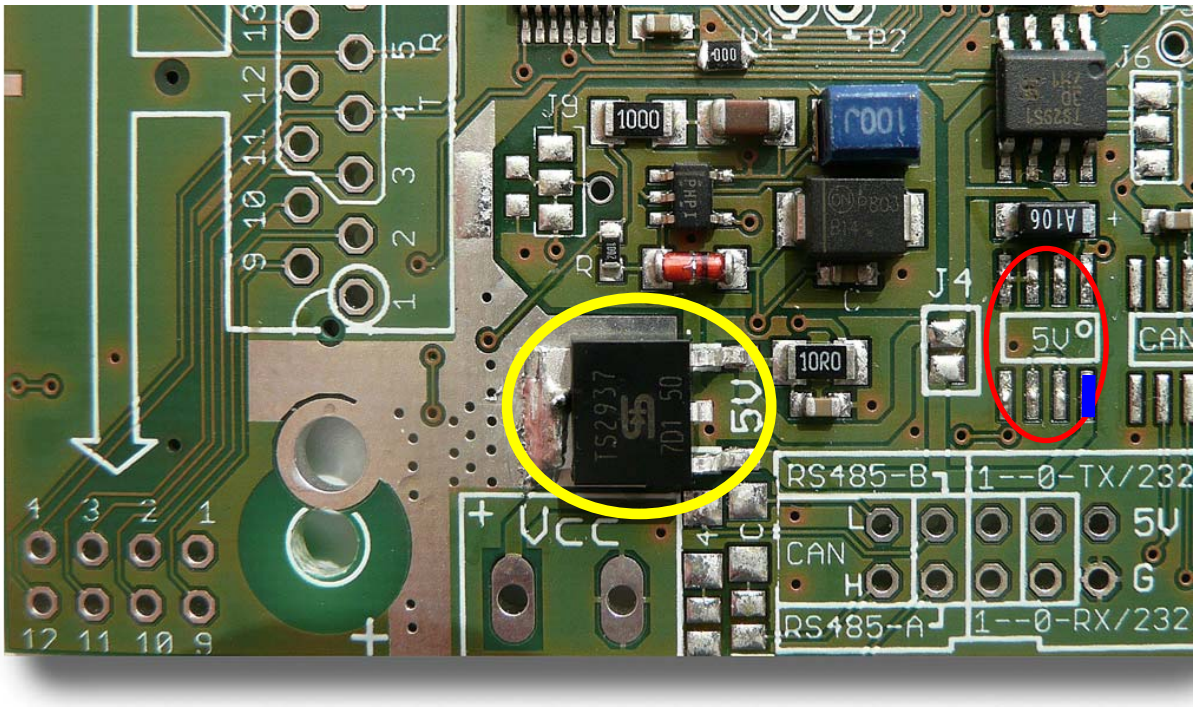
The default condition as delivered to you is: 5 Volts regulator present, Controller, RS232, RS485, CAN, Lighting-booster are running with 5 Volts.

“Bigger” voltage regulator

On our board, we are using linear regulators which will burn the overvoltage into heat. In opposite there are switched regulators available, but these are more complex and will use much more (not available) space on the board.

The default 5 Volts regulator on our board comes in a SO8 package (see photo, red circle) and allows a continuous input of maximal 12 Volts. Due to the missing cooling capacities it cannot burn more voltage.

To allow the optional running with a larger voltage than 12 Volts, there is a position on the board available to solder in a larger regulator. This one offers some heat sink option and thus allows a continuous running up to approx. 18 Volts. The positioning of this regulator (SMD TO252 package) is showed in the yellow circle.



If you solder in a larger regulator, you shall desolder the default regulator (see red marking). Alternatively, you may desolder only pin 8 of the regulator (see blue marking) and bend it upwards so it does not have a connection to the pad. It is not recommended to run two regulators in parallel – at least one of them will be damaged.

Some theory on heat sinking

Why is the small default regulator not being able to drive the board with 14 Volt at 70mA, whereas the datasheet of the regulator shows a maximum of 150mA? Well, some theory on this:

Assumend you connect 14 Volt, the regulator shall output 5 Volt and the schematics need 70mA. Then, we have a power dissipation P_D of (14Volts – 5Volts) x 0,07A = 630mW.
Also the loss through the regulator itself has to be taken into consideration (current through GND generally approx. 12mA) – thus 14 Volts x 0,012A = 0,168mW.
The total power dissipation P_D now comes up to approx. 800mW.

Assuming that we allow the maximal junction temperature of 125°C and we assume a maximale ambient temperature of 40°C, we now get a thermal resistance T_R(max) of 85°C.

$\theta_{(j\text{-}a)} = T_R(max) / P_D \rightarrow$ 85°C/0,8W = 106 °C/W

The regulator in the S08 package allows 160°C/W or more according to the specs. As soon as the calculated value $\theta_{(j\text{-}a)}$ falls <u>below</u> this 160°C/W, the regulator get too hot and will eventually run into the emergency shutdown. Any cooling is not possible at a S08 package.

As of the calculation above it is not possible to run the D071x module with 14 Volts and the default S08 regulator. Some more math shows that is will be allowed to run at:

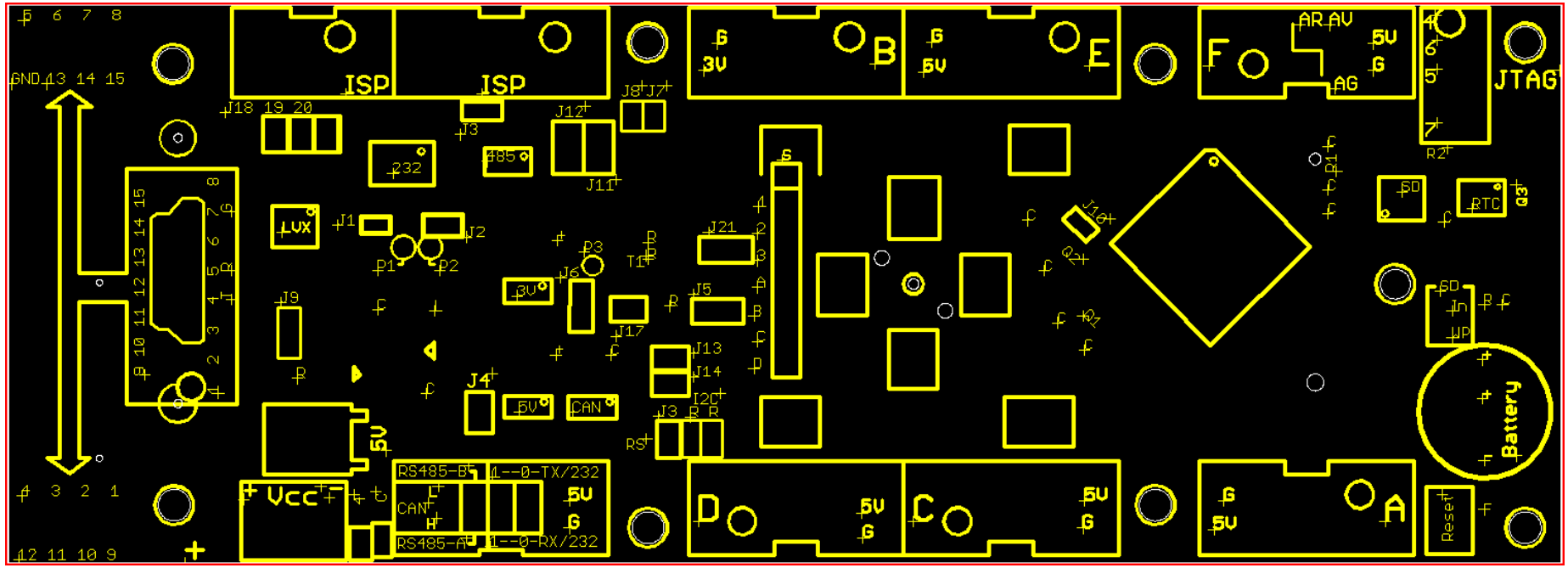
- 14 Volts and 70mA and a maximal ambient temperature of -3°C
- 14 Volts input and a maximal continuous current of 44mA
- At 70mA as maximal input voltage of 11,2 Volts (Remark: as there is a minimal cooling through the soldered pads, 12 Volts will be barely possible).

<u>Alternative 1:</u> a larger regulator in a TO220 package (such a 7805) allows 60°C/W; a TO-263 package anyhow 80°C/W – without external heak sink. This could be an alternative. Unfortunately, these are no SMD parts and will take much space - too much space for our board.

<u>Alternative 2 – using PCB copper as a heatsink:</u> An alternative small SMD regulator in a TO252 package allows (without a heatsink) a thermal resistance of 150 °C/W. This value is not a big improvement to our default S08 regulator. But this is valid only at the first glance, as this TO252 regular offers a small heatsink which we can solder to the PCB and use some PCB area as an expanded heatsink. As a rule of thumb, you may assume 5cm² of PCB copper (without solder resist!) decline $\theta_{(j\text{-}a)}$ from 150°C/W to approx. 50% .
On our PCB of the D071x, we are using both sides of the PCB with many vias as thermal connections as a heatsink. We are estimating a thermal resistance of approx. 90-100°C/W. Thus, at maximal 40°C ambient temperature approx. 16 Volts input are allowed; at 25°C ambient temperature approx. 18 Volts are allowed.

Connection pads and jumper:

The placement of the connectors and the jumper positioning is evident from the picture below.



To simplify the work with the external modules, we did place GND and 5V at each connector block. **Exception:** Connector of PortB: here we are offering 3 Volt and GND.

The pins at the shrouded headers are all used in an identical way: (exemplary showed for Port E)

Pin 1: PortE 0	Pin 2: PortE 1	Pin 3: PortE 2	Pin 4: PortE 3	Pin 5: PortE 4
Pin 6: PortE 5	Pin 7: PortE 6	Pin 8: PortE 7	Pin 9: GND	Pin 10: Vcc

Hint: If you want to connect with other devices for sending / receiving data, you shall read the chapter „termination“ at the supplement of this manual.

Several options of this board will use some of the existing ports. These ports may not be usable for you anymore. Please check on details at the end of this document (see page 60).

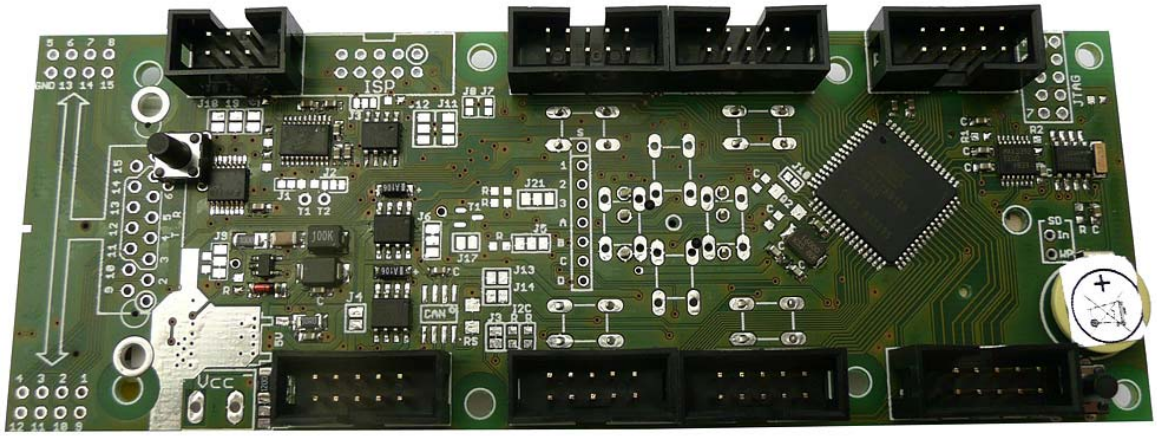
Variance: The connector for Port F and Port A need to be soldered 180° turned around (different from the other, the opening of these two connecters shall be turned to the inside). At the connector of Port F only F0-F4 are present, the other pads are being used by ARef, AGND and AVCC. If you need Port F5-F7, you need to contact these at the JTAG connector alongside (the pads F4-F7 are available there and are marked with „*4, 5, 6, 7*“).

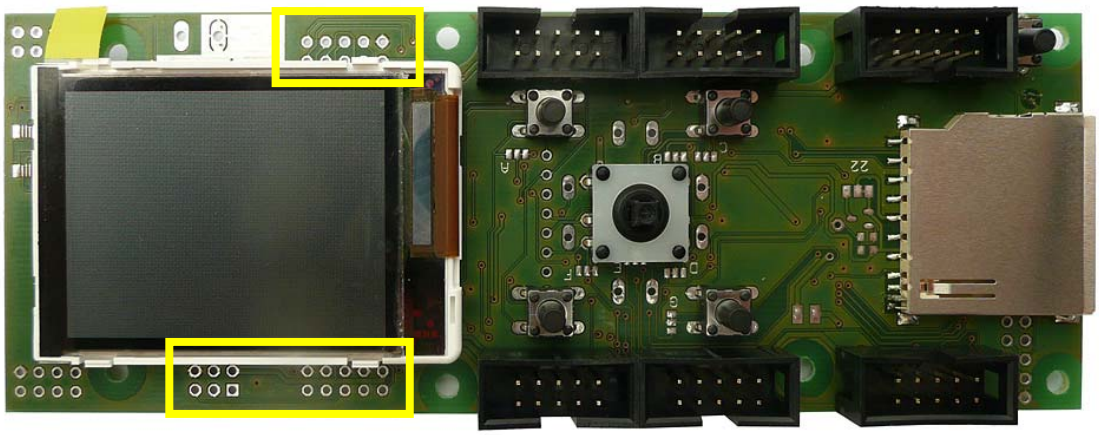
The meaning of the several jumper, their positioning and the default delivery, you may gather from the table “jumper allocation” at the end of the document at page 60.

Connectivity options

Shrouded headers

The board is prepared for shrouded headers. These can be soldered from the bottom side and mostly (not all) from the display side. As this module usually is being placed behind a front plate or in housing, the preferred mounting direction is the part side of the PCB (as showed at the upper photo below).

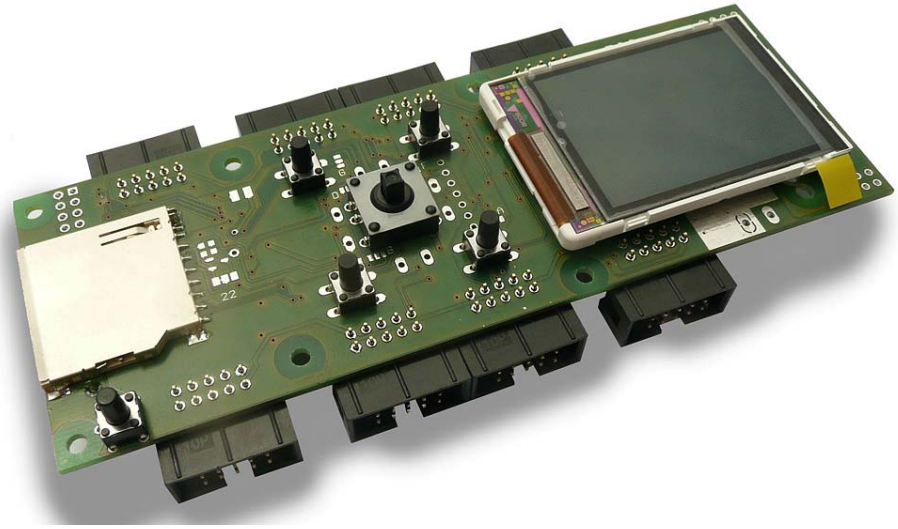


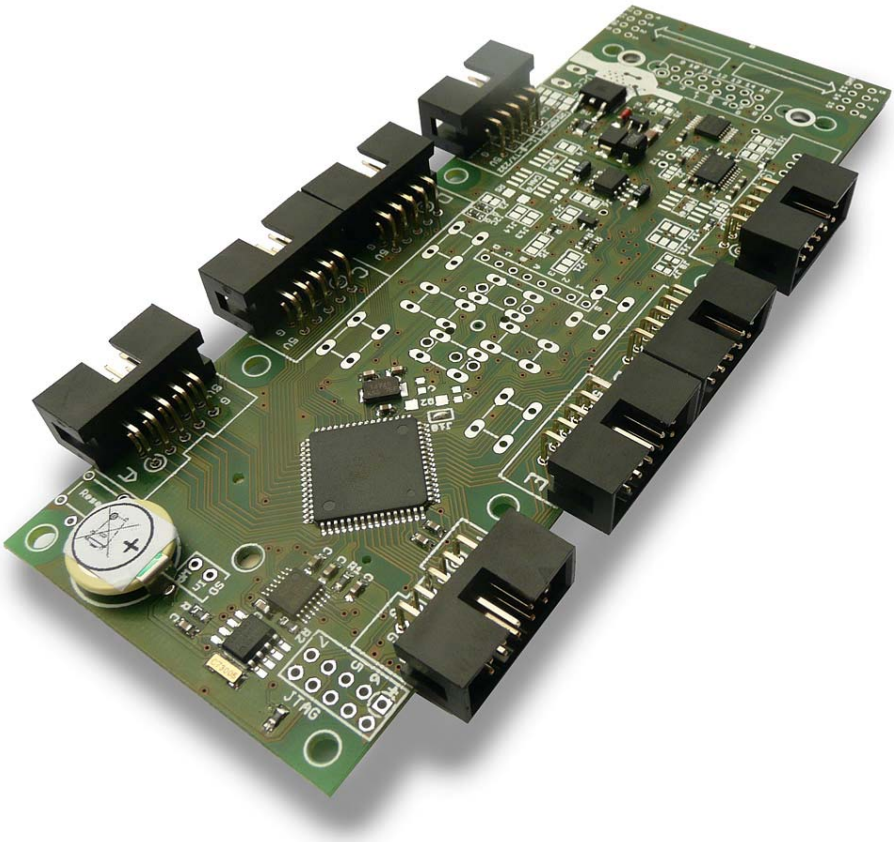


Placing them on the display side is also possible, but the space around the display will be enough for a shrouded header.

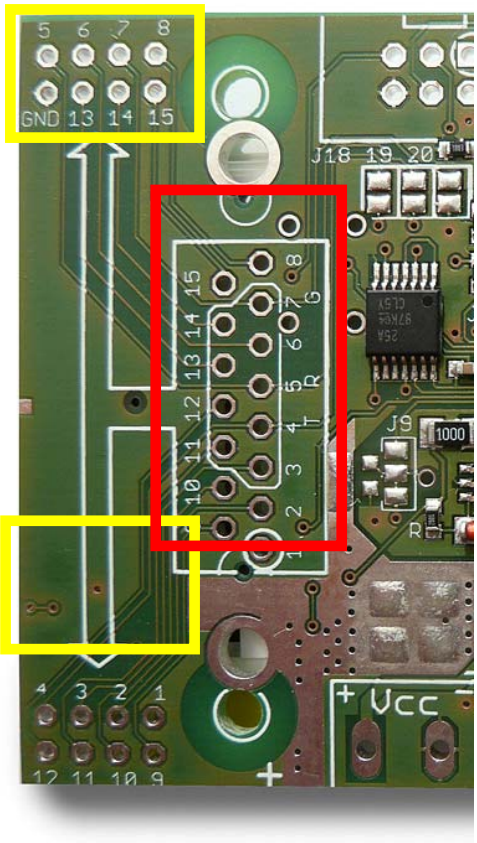
Caution: If you want to set up the PCB in our housing BS600: The housing with battery cage (BS601) does not allow the usage of shrouded headers at all positions as you will not be able to close the housing then. Please check the positioning first before soldering the connectors in.

Other alternative: You may also use 90° angled shrouded headers. Please make sure then that the connector for port F and port A are different as of the 180° rotated allocation.





15-pin standard DSub-connector



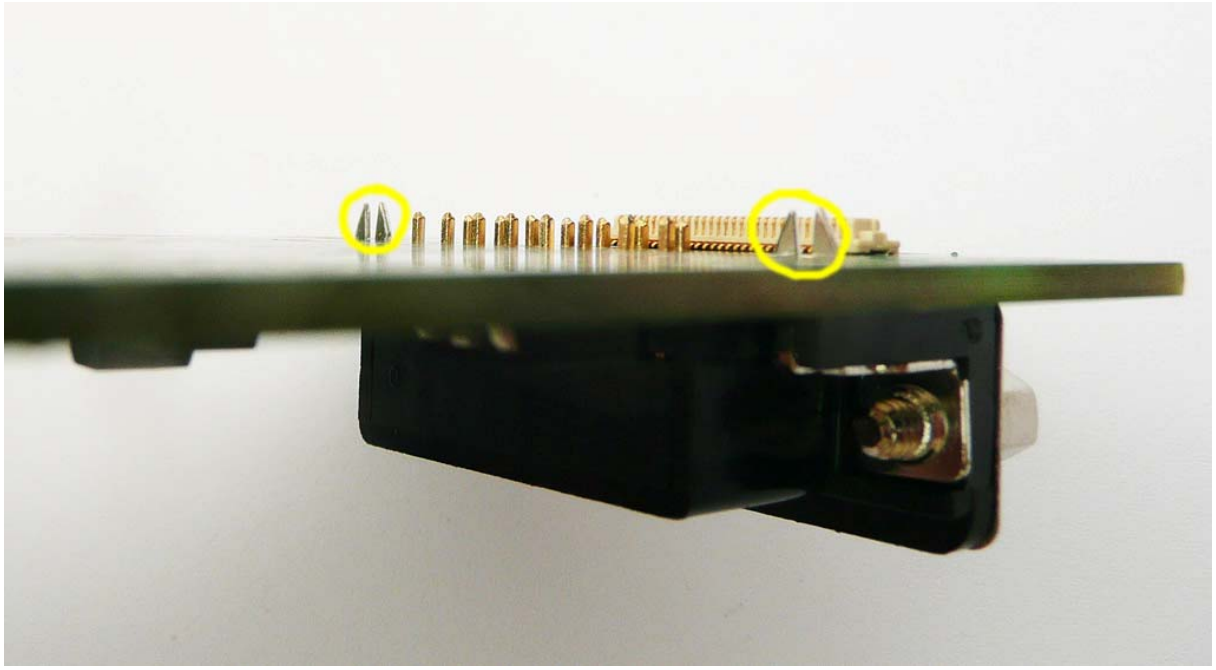
Remark: The connector can be a male or female type as you need.

At the left area, the PCB offers the option to use a DSub connector (see red mark). This connector is NOT connected to any ports but to patch-pads. You will find above and below the connector patch-pads which numbering are identical to the connector itself. From these patch-pads (yellow mark), you may set up a connection to any needed port.
E.g. if you want to connect Port A5 with pin 9 of the DSub connector, you just link a short wire from Port A5 to the patch pad marked with a "9".

Optional you may connect Tx and Rx of the RS232 interface UART1 directly to the connector (see chapter "9Pin D-Sub Connector" next page).
These two signals and GND are already prepared.

Shortening of the clips

The two clips of the Dsub connector are probably to long and might disturb the correct placement of the display on the opposite side of the PCB. In this case, you should shorten these clips with a wire cutter. The soldering pins of this connector are usually short enough and will not disturb the display.

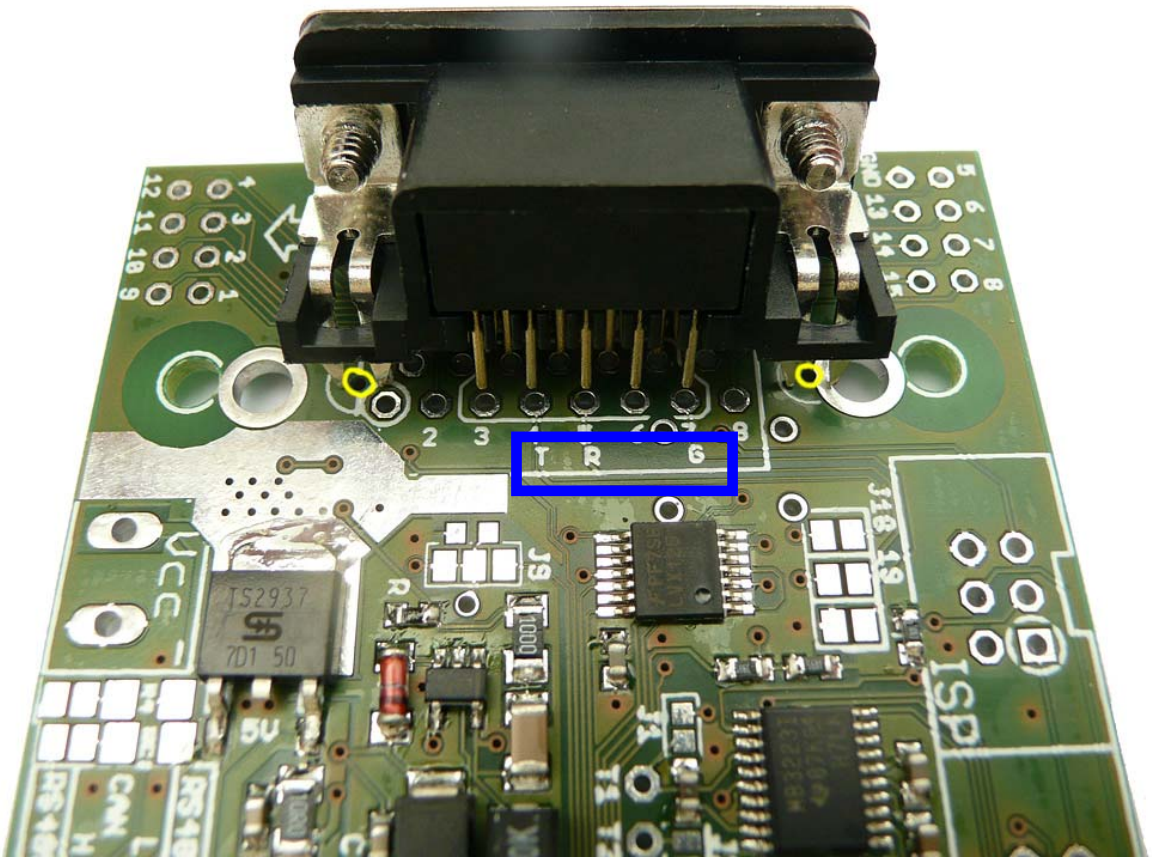


9-pin DSub connector

Optional you may use a 9pin Dsub connector. For the correct positioning please check the printing on the PCB (see photo) – you shall use the pins 3, 4, 5, 6, 7 of the outer row. This 9pin connector offers the advantage that the RS232 signal is connected to the correct pins to allow a direct connection to a PC by just plugging a standard serial cable to the PC and the module. The high level RX, TX and GND signal of UART1 will be connected to the Dsub connector by closing 3 jumpers on our board (see chapter "*Connecting Dsub connector to the serial interface UART1*" later on). At the pads marked with „T", „R" and „G" (see blue mark at the photo), the signals TX, RX and GND are now present and may be connected to a PC or other device with a RS232 interface by just using any standard cable.

One compromise is needed to be able to use a 9-pin Dsub connector instead a 15pin type: We could not prepare the larger hole for the fixing-clips of the 9pin connector, as these would be needed in an area where soldering pads of the 15pin connector are placed. If you are sure that you will not need the 15pin connector, you are able to drill the two holes for the 9pin connector with a 3mm drill. To make it easier for you, we already marked this area and we predrilled the hole with a small diameter (0.8 mm) for you (see the yellow circles at the photo). Please notice the section of shortening the mounting clips on the preceding page.

Hint: Instead of drilling you may also completely cut away the mounting clips of the connector.

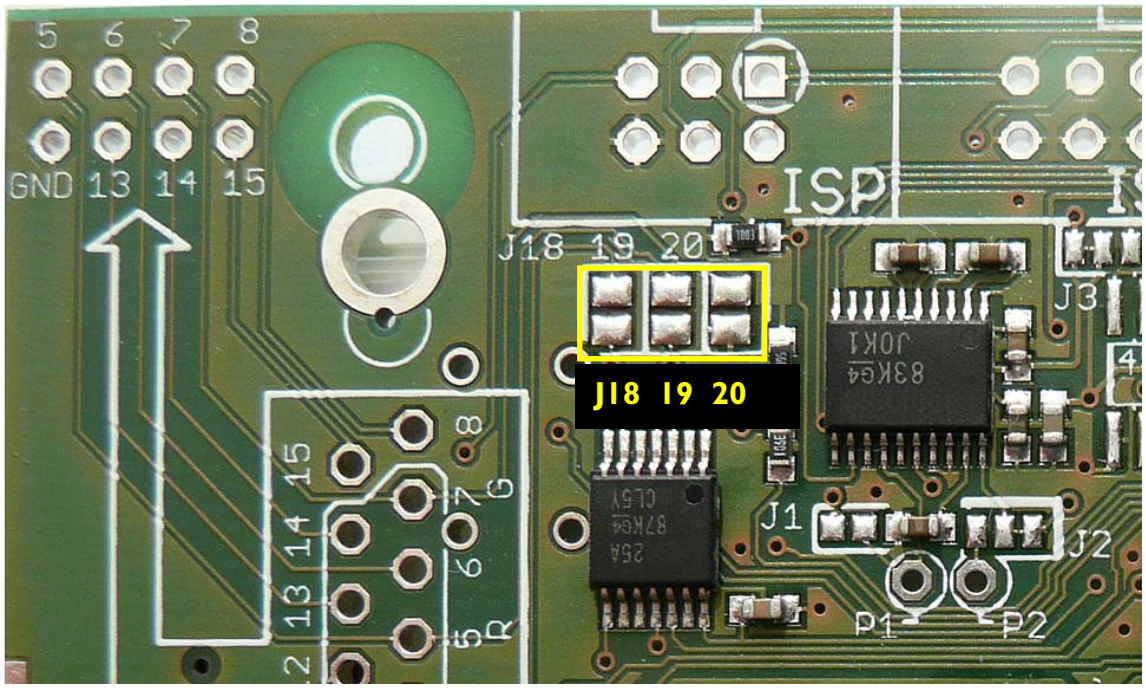


When using the 9pin DSub connector, the numbering scheme at the two bridge connectors at the upper and lower side of the PCB are not correct anymore. You then need the following table for the correct usage.

Writing on PCB	Usage at D-Sub-15	Usage at D-Sub-9
1	1	-
2	2	-
3	3	1
4 (Tx – see next chapter)	4	2
5 (Rx – see next chapter)	5	3
6	6	4
7 (GND – see next chapter)	7	5
8	8	-
9	9	-
10	10	-
11	11	6
12	12	7
13	13	8
14	14	9
15	15	-
GND	-	-

Connecting of the DSub connector with the serial Interface UART1

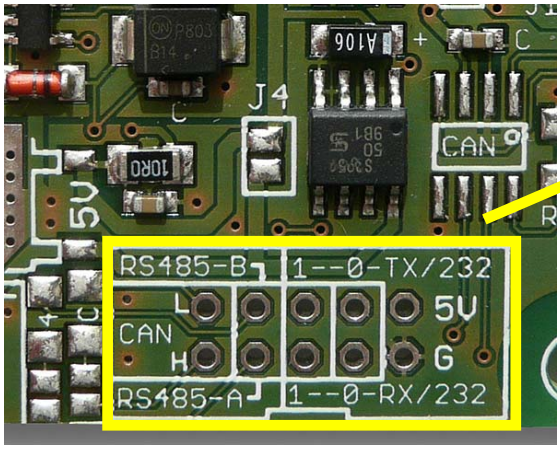
Just three Jumper (No. 18, 19 and 20) shall be closed to connect the DSub with the high level RS232 signals of UART1.

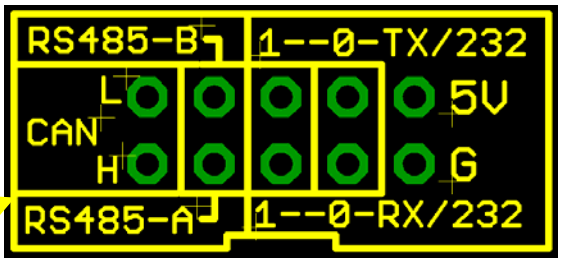


The connectors for other interfaces (RS232, RS485, CAN)

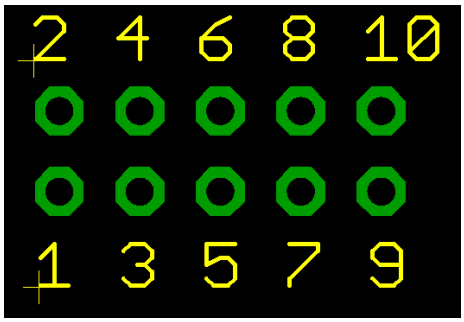
On the right side beneath the VCC connector there is a connection area for some of the optional interfaces:

- UART0 (RS232 High Level Rx, Tx)
- UART1 (RS232 High Level Rx, Tx)
- RS485 (A & B)
- CAN Bus (H & L)





To give you a better overview (the printing on the PCB at this area is bit crowded due to the limited space), we are showing the connector usage in the following table:



Pad	1	2	3	4	5	6	7	8	9	10
Usage	CAN H	CAN L	RS485 A	RS485 B	RS232 RX1	RS232 TX1	RS232 RX0	RS232 TX0	GND	5V

Please keep in mind that these connector pads may be connected to negative voltages and also voltages with +12 or -12 Volts. Any careless made connection from one of these signals to any other part of the PCB may damage or destroy the board or parts of it. You shall only use these pads to connect them to external signal lines.

Options for the human interface

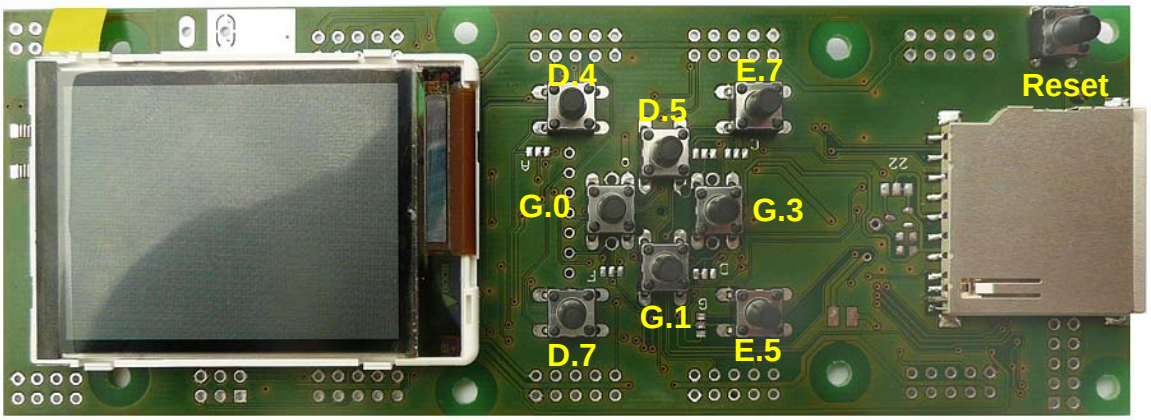
Push button switches

The switches of the module are already preconnected with several ports (usually PortD and Port G) and will pull the appropriate line to low (GND). If you want to use the switches or the optional membrane keypad, make sure the used ports for the switches are defined as **INPUT ports** and not as output ports. As these lines are pulled against GND by pressing a switch, any port defined as a output port would be short circuited and thus destroyed.

The correct approach would be:

1) Define switch port as input
2) Switch on the internal pull up resistor

Thus the switches are having a high level signal (through the pull up resistor). By reading the port it shows a high level (1) when the switch is not pressed. If some presses one switch the signal is pulled against GND and the reading of the port returns a low (0) signal. As the current through the pull-up resistor is very small, there is no danger.

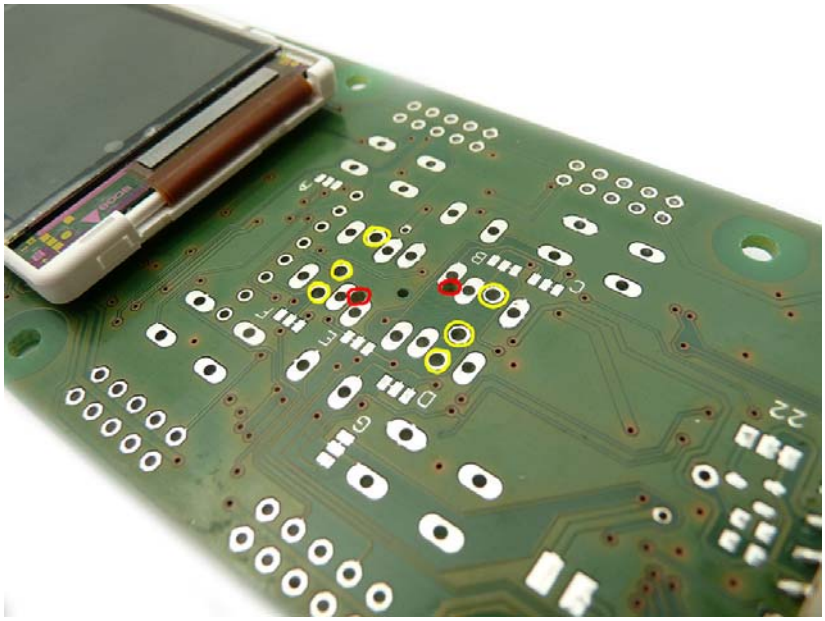


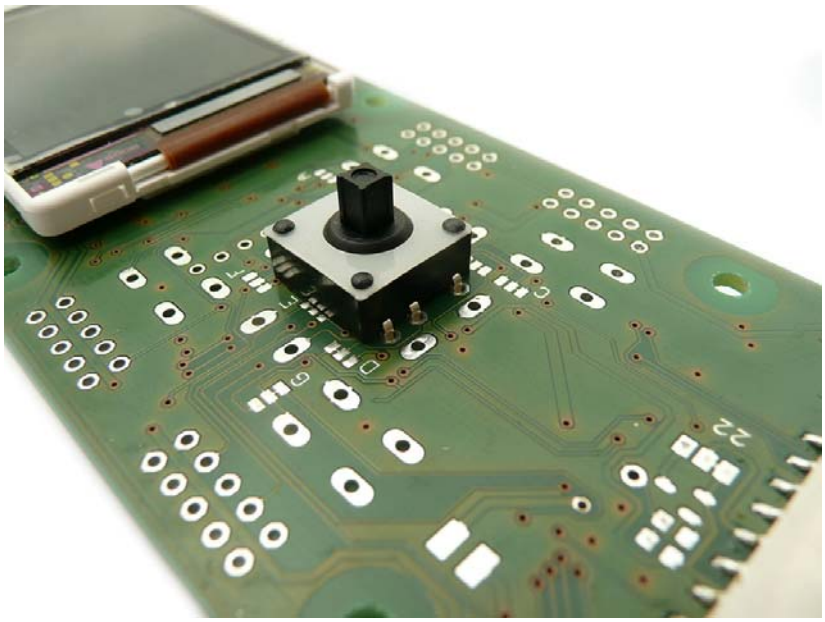
The pre connected ports are shown at the printing above. This is the delivery configuration. It might happen that you need one or more of these ports for another task. We prepared our board for this. Most switches can be assigned to a different port. More on this can be found at the end of this chapter (page 27).

Hint: The switches do not need to become soldered (at least at the beginning). The are well fixed by just pushing them into the soldering pads. Doing this, you may easily try out which switches you really need and which may be removed.

Joystick

Instead the centered four switches, you can use a joystick. The usage is identical to the regular switches (e.g. the "up" switch uses port G0). The joystick soldering contacts are shown (yellow markings) in the picture below. The red markings are presenting the positioning pins of the joystick. The joystick as well can be used at the beginning without soldering. Just pushing it into its place keeps it firmly in position.





As the joystick is using the same ports as the four centred push button switches, you may reconfigure the port usage for the joystick as well. Furthermore the joystick offers a 5th possible usage (beside the four directions): a push function which is connected to port D6 by default. Together with the joystick, you may use the other available four push button switches around the usage as additional user input buttons.

Membrane keypad

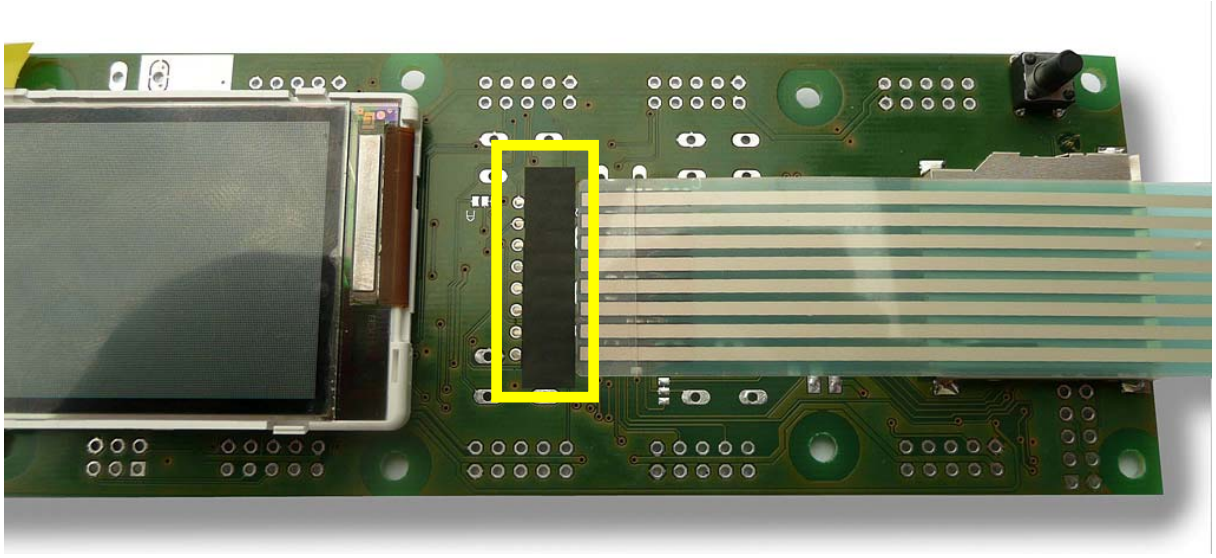
Especially for the housing BS600 and BS601, we developed and produced a multi function membrane keypad.

By the way: If you need at least 50-100 keypads, we are able to produce them especially to your needs. Starting from 30 keypads, we can change just the printing to provide you, an individual keypad with your logo on it, different printing on the keys etc... Just ask.

For the connection of the keypad, just solder in the FPC connector as shown below. The FPC cable of the membrane keypad will then just be pushed into the connector.

For the 8 wires of the keypad, we are using the description S, 1, 2, 3, A, B, C, D.

The picture below shows the wire "S" at the lower end and the wire "D" at the upper end.



D
C
B
A
3
2
1
S

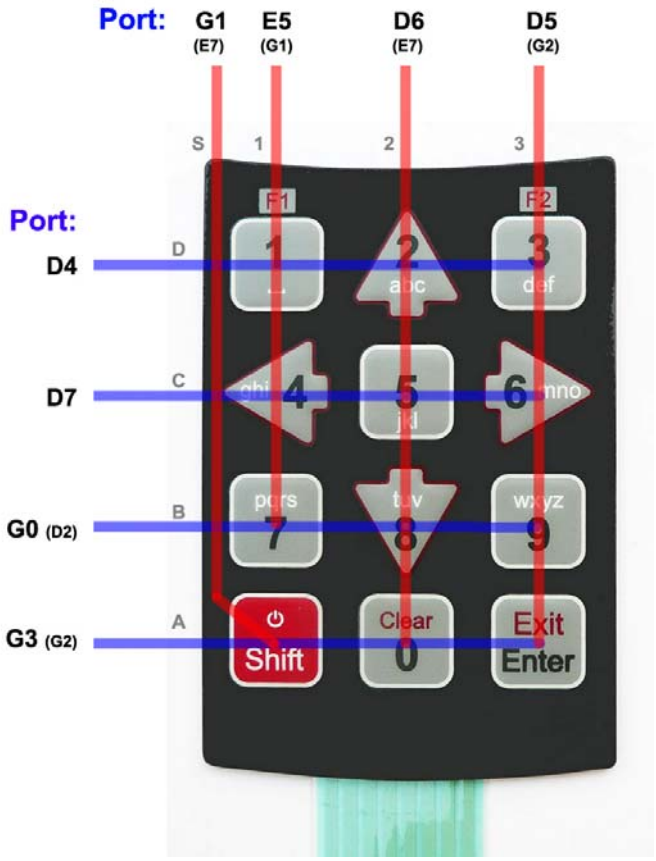
The allocation of the membrane keypad as a matrix has been done as follows:

	S	1	2	3
D		1	2	3
C		4	5	6
B		7	8	9
A	Shift		0	Enter

Why are we using a matrix? Without a matrix, we would need 12 ports for the used 12 switches. Now by using a matrix, we only need 7 ports. For this keyboard, we are using a 8th port on purpose as we wanted to connect the red button (S) to its own individual port (for a switch on/off usage).

Polling the matrix:
Reading out which key is pressed is a bit more complex than checking switches which are using a single port line each.

For a better understanding, we are showing the keypad with its matrix connection alongside. The port used by default has been printed in bold, the alternative usage (changeable by a jumper as explained after the next page) is printed below /beside in a smaller font.



Following is a sample how to read out a pushed key:

1) All ports used for the switches has to be switched to input. Caution: do not switch to output as you may destroy the port then. The pull-up resistors of all used ports has to be switched on.

2) Now you have to reconfigure row D (thus port D4) as an output and you set the output to low (0).

3) Consecutively, you now check the input ports of the columns 1, 2, 3 (thus Port E5, D6, D5). If one of these wires shows a "0", you know the pressed key from the matrix. The column "S" (Port G1) only needs to be read out if row "A" is switched to low, as this is the only possible connection for the red key. Remember setting back the output port back to input and to switch on the pull up resistor after reading out the three columns.

4) The same (configuring a row as output and reading the columns) applies then for the remaining rows C (port D7), B (port G0) and A (port G3) before you start over again with row "D"

Examples:
Row D: port D4 to output and set to low; Reading out input ports (Pin) E5 (= column 1) shows a „0": This means the keypad „1" has been pressed, as usually input port E5 shows a high signal. If it is low the input port with its pull-up resistor port has been pulled down against GND. After reading out: switch back port D4 to output and switch on the pull-up resistor.

Row C: port D7 to output and set to low; Reading out column „1" is showing a "1": no key is pressed in this column. Reading out column "2" shows a "0": the keypad "5" has been pressed. After reading out: switch back port D7 to output and switch on the pull-up resistor.

Row B: port G0 to output and set to low; reading the columns 1 to 3 (E5, D6, D5) is showing: all wires are high = no switch has been pressed. After reading out: switch back port G0 to output and switch on the pull-up resistor.

Row A: port G3 to output and to low; Checking column S (port G1) is giving a „0"; also column 3 (port D5) shows a „0": in this case the key „Shift" and also the key „Enter" has been pressed – the user might want to Exit (Shift-Enter = Exit). After reading out: switch back port G3 to output and switch on the pull-up resistor.

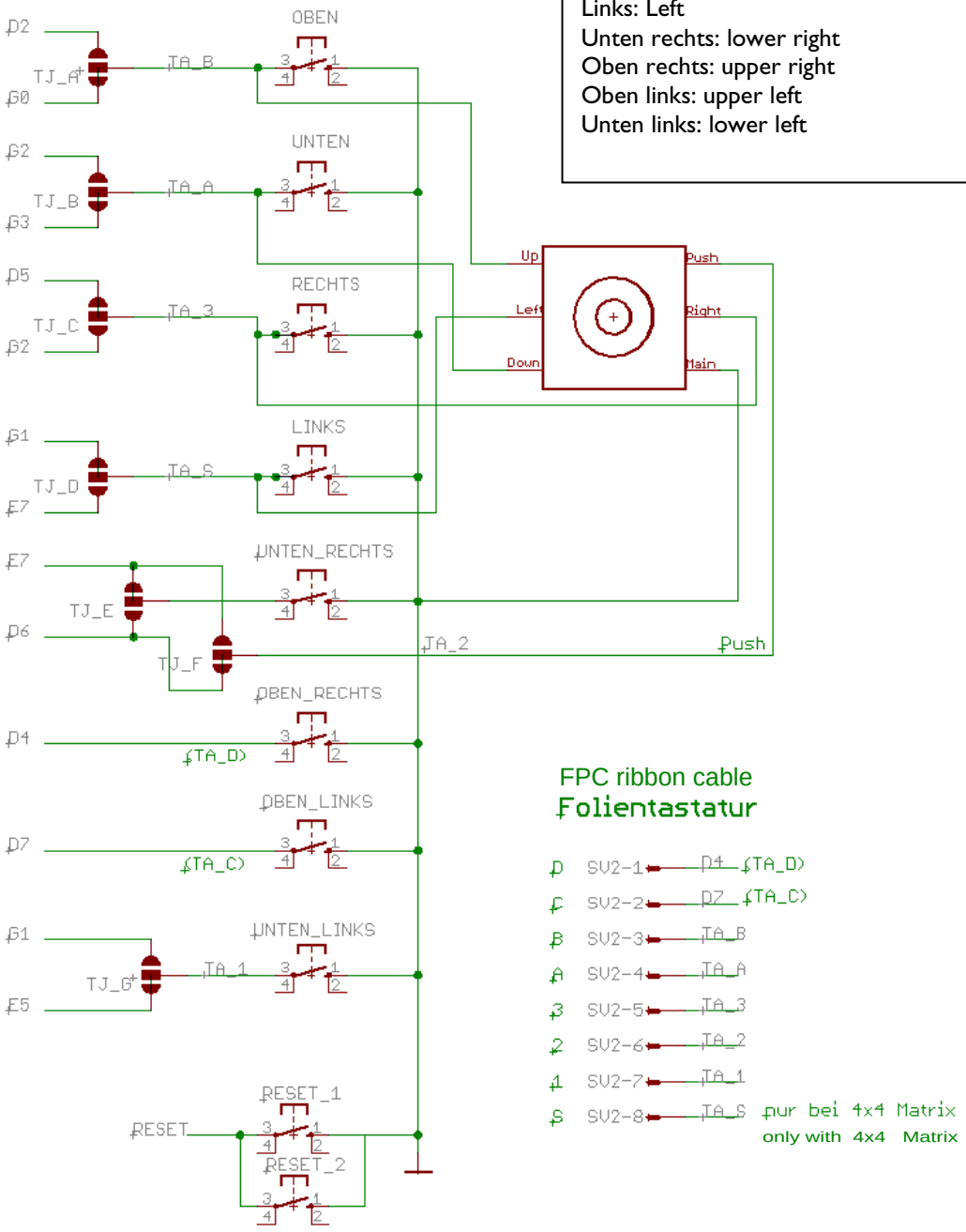
Summary: The ports switched to output are responsible to pull the column wires to low if a key has been pressed. This level change can be detected.

You shall never define any key-switch port to output and to high, as then, by pressing one or more keys you may short circuit a port and then destroy the port and possibly the switch.

A high level at the ports shall only be created by setting these ports to input and then by switching on the internal pull up resistor. By this the current is very low if a port is been pulled to GND level.

Schematics of the switches area:

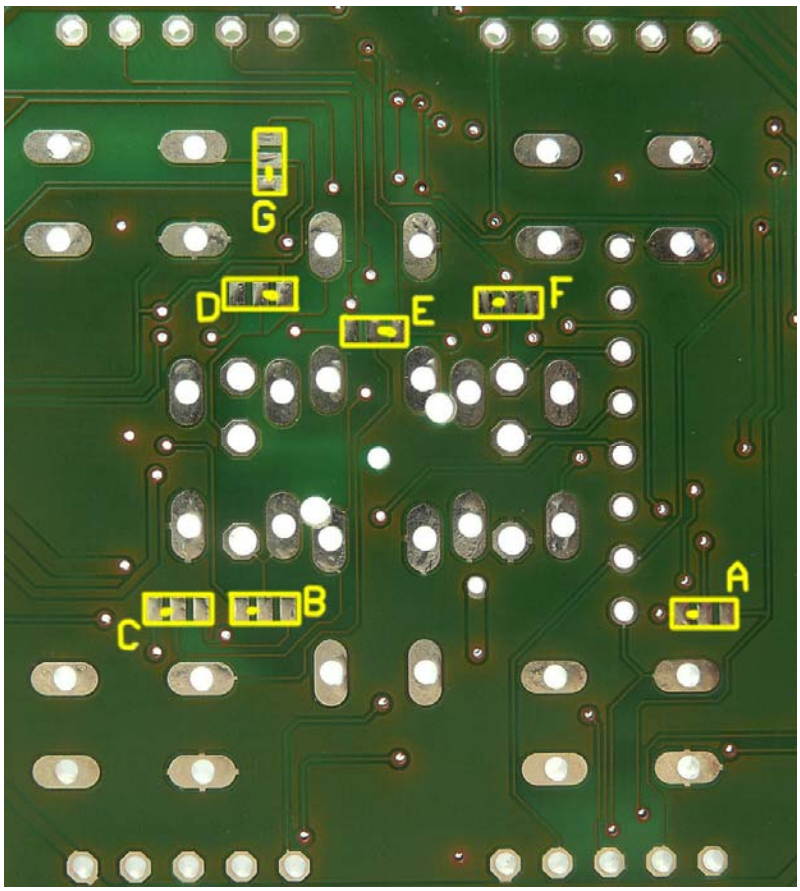
Translation of the used printing:
Oben: up
Unten: down
Rechts: Right
Links: Left
Unten rechts: lower right
Oben rechts: upper right
Oben links: upper left
Unten links: lower left



The default usage of the jumper can be gathered above (you will see the default connection inside jumper). For example the jumper TJ_A may connect the switch either to port D2 or port G0. The default connection is port G0.

The switches *up, down, right, left* (Oben, Unten, Rechts, Links) equal the centred switches AND the joystick directions.

Reconfiguration of the used switch port



Under certain circumstances, you might need one or more ports we are using by default for the switches for other tasks. For this event, we prepared the board so you may reconfigure most switches to a different port.

With the switch-jumpers TJ_A to TJ_G you can reconfigure the switches. You will find the jumper and their writing at the upper (display) side of the PCB. These jumpers are two-way-jumpers where the middle pad has to be connected with one of the adjacent pads. If you want to change the default connection you need a sharp knife (scalpel) which you use to cut the default closing wire between the middle pad and the default pad. Then you use your soldering iron to close the connection between the formerly unused pad and the middle field with a drop of solder. The default connection of the jumper is marked yellow above.

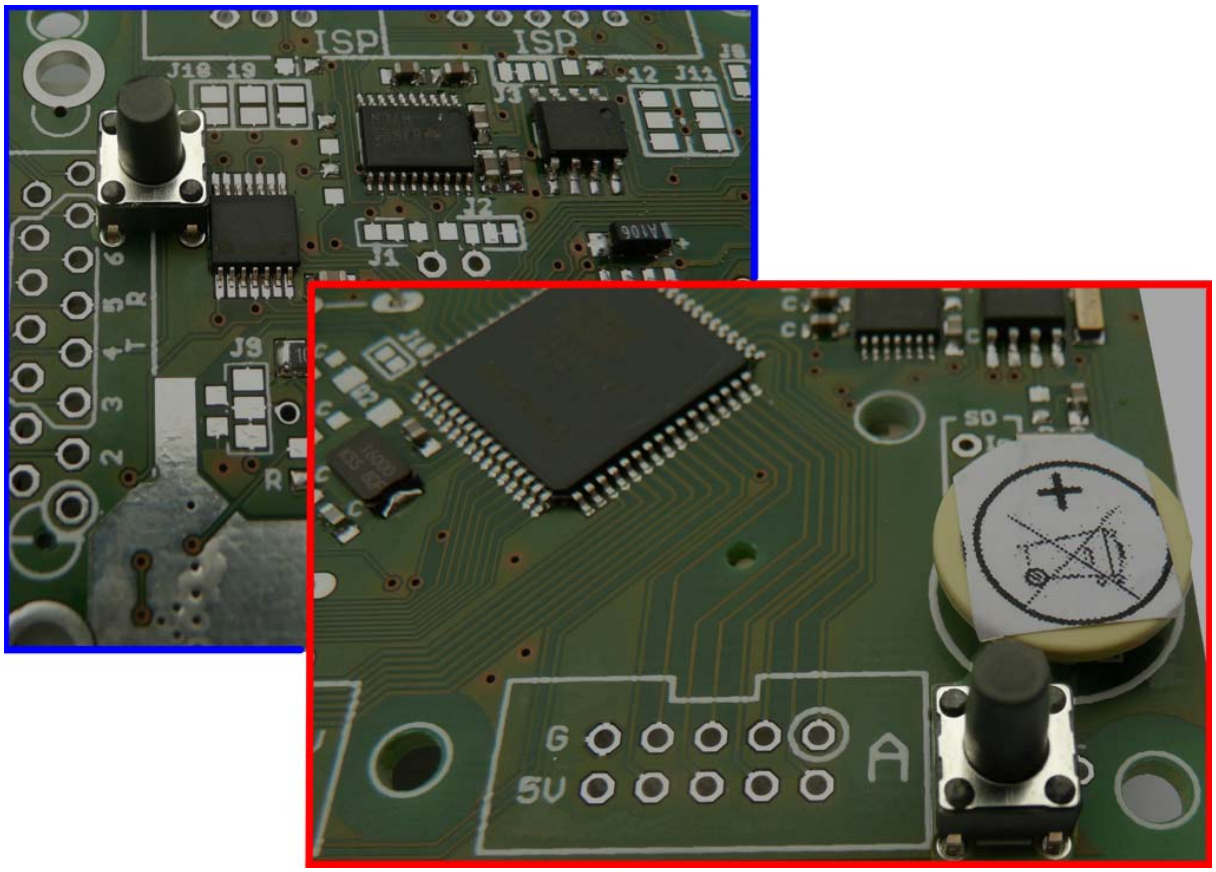
It is not possible to have alternative ports for every switch at the same time. Some alternative ports are connected to more than one jumper. If all fails to your needed connections which are not prepared yet, you may make your own connection by soldering a small wire to the middle field of the jumper and then connect it to whatever port you like (in this case the jumper shall not be closed in any direction)

Switch	Membrane pad (row / column)	Default port	Possible alternative port	Responsible Jumper
Up	**Row B**	G0	D2	TJ_**A**
Down	**Row A**	G3	G2	TJ_**B**
Left	**Row S**	G1	E7	TJ_**D**
Right	**Column 3**	D5	G2	TJ_**C**
Joystick: Push	**Column 2**	D6	E7	TJ_**F**
Upper left	**Row C**	D7	–	–
Upper right	**Row D**	D4	–	–
Lower left	**Column 1**	E5	G1	TJ_**G**
Lower right	**–**	E7	D6	TJ_**E**

Reset-switch

The board D071x offers two possible positions for a reset switch. The first position is located in the lower right area beneath the pads of port A (red framed photo). This switch may be used from both sides of the PCB. Upper (display) side: available by the user from the main interface area. Lower (part) side of the PCB: if the boards sits in a handheld device, the reset key can be accessed e.g. through a small hole at the bottom of the housing (e.g. a straightened paper clip).

A second, alternative position is available on the left side (blue framed photo). Here, the switch can only be accessed from the bottom side (e.g. through a housing as explained above) as the display is located on the other side of the PCB. Please note that you might need to pinch off some of the switch wire on the display side of the board as they are able to disturb the correct positioning of the display.

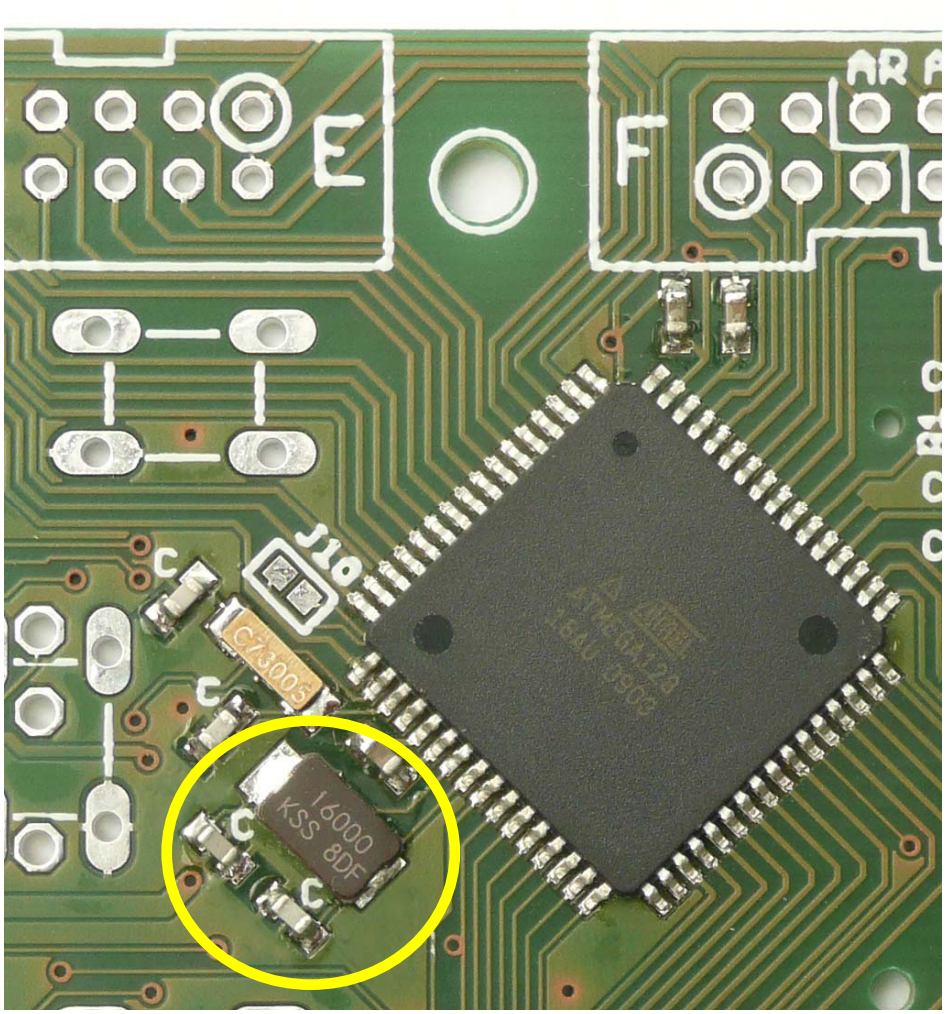


Clock / Crystals

System clock

The microcontroller comes with 8 Mhz clock speed as default (if you did not order it including the upgrade to 14.7456 or 16 Mhz).

If you like you may enhance the speed of the microcontroller by soldering 2 x 15pF crystals (in SMD size 603) and a SMD crystal to the PCB. Caution: the SMD crystal shall not be larger than 3x5 mm as of the limited space on the board.

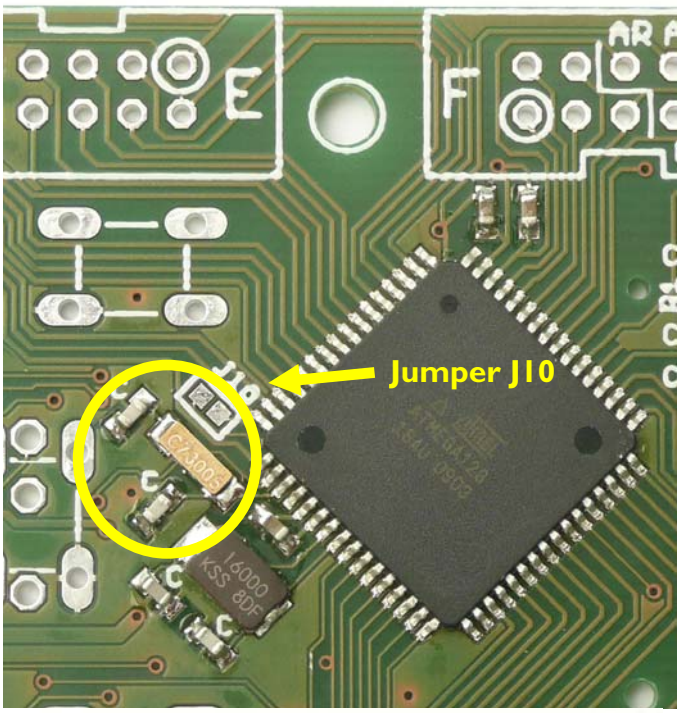


If you are finished with this, do not forget to change the fuse bit for the system clock from 0100 (8 Mhz) to 1111 (external high speed crystal). **Do this only <u>after</u> you added the crystal as explained above.** Any other fuse bit setting or a missing crystal will result in a non working microcontroller.

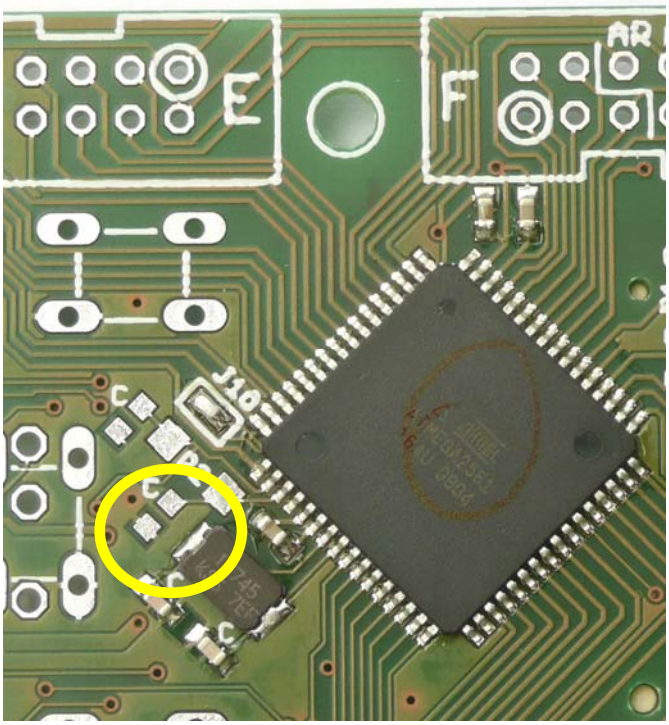
Do not play around with the fuse bit settings. Wrong settings can disable your microcontroller. Please: if in doubt: send a mail or call us!

Clock crystal for the internal Real Time Counter

If you like to use the internal Real Time Counter (do not confuse with the Real Time Clock, both abbreviated with RTC), you will need an external clock crystal or an external clock signal.
Our board is prepared for a 32.768 Khz SMD clock crystal (maximum size of 1.3 x 4mm) as well as two needed 15 pF SMD capacities (size 603).
The real time counter can be used for adding a separate clock for internal usage (e.g. for periodically waking up the microcontroller in sleep mode or to realise an internal watch).



If you ordered the real time clock AND a ATMega2561 or AT90CAN128, you may spare the clock crystal: the real time clock is able to provide a 32.768 Khz clock signal to the real time counter in the microcontroller. For this you just have to close jumper J10 with a drop of solder (if you ordered the RTC with the board we usually do so during production).



In this case you need to the Atmel controller, that the real time counter shall not use a crystal but an external signal: Bit EXCLK at the register ASSR needs to be set to "1" (e.g. Bascom: „*Set Assr.exclk*").

Caution: According to the data-sheet, the real time counter of an ATMega128 shall not be driven by an external signal and needs its own clock crystal. Thus, avoid closing J10 on a ATMega 128.

Remark: If you want to use the clock crystal (connected to TOSC1 / TOSC2), you will not be able to use port G3 anymore (as this is TOSC2). Then you need to reconfigure the key switch jumper (jumper TJ_B) for the "down"-switch from G3 to G2.

Data Interfaces

For the communication with other devices the board is offering 2 x RS232 interfaces, a RS485 interface and a CAN bus interface as options. Additionally there are I²C and SPI available.

SPI Bus:

The SPI Bus is a serial high speed bus which is being used by the display and the SD card module on our board. There are three common lines being shared by any device using the bus:

- Data In
- Data Out
- Clock

as well as other needed lines depending on the used device. Any device at the SPI bus uses its own individual CS line (chip select) where its being told if the signals on the bus are meant for it or shall ignored (as meant for another device).

SPI allows a very quick data transfer, but only on very short wires (several cm).

The used Atmel microcontroller uses a hardware SPI which allows maximum speed. For hardware SPI the shared ports to use for the bus are predefined and cannot be changed.

Used:

- Port B1: Clock line
- Port B2: Data Out (MOSI)
- Port B3: Data In (MISO)

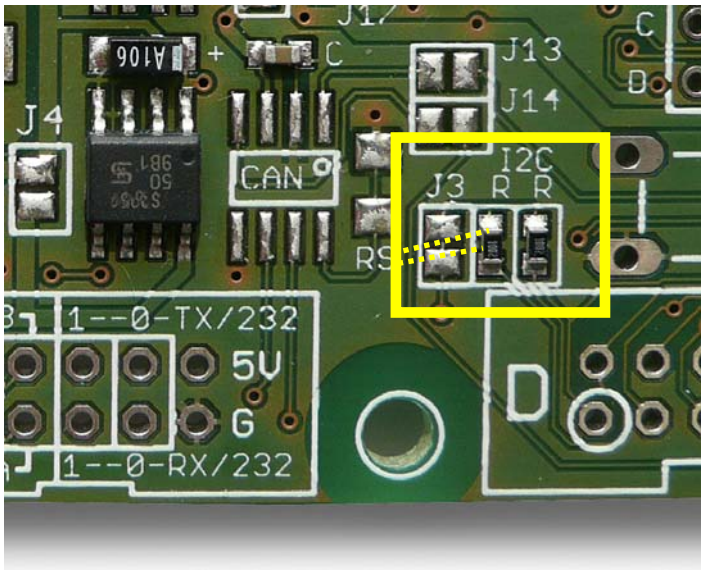
These three ports shall not be used for any other usage (but adding other SPI devices).

I²C / TWI – Two Wire Interface

The ATMega microcontroller offers another interface called TWI (two wire interface), better known with the acronym I²C (speak: I square C) which was invented by Philips. Just for license reasons Atmel is calling the interface TWI instead I²C..
This interface needs only two wires and a common ground. TWI is a synchronous two wire bus where one wire transports the clock signal and the other the data signal.
What is it, used for In many modern system several devices need to communicate to each other. By connecting them all a lot of wires would be needed. A simple bus system allows to use them all on the same wire and communicate either with each other or with a master unit. The big advantage of the I²C–bus is its simplicity. There is no predefined speed which makes it easy to connect quick and slow devices to one bus. More on this theme can be found in the world wide web.

TWI (=I²C) is available at the Atmel microcontroller at port D0 and D1. This means, you shall not use these two ports as they are used for the data traffic of all I²C devices (if any).

The I²C bus needs both lines pulled up with a pull up resistor. Usually this happens at the master bus module. We did prepare this pull up resistors by default on our board. If there are any other master modules on the bus, you need to remove all pull-up resistors but one. On our board there is no need to desolder anything: just open a prepared jumper to remove the resistors from the bus.



The yellow frame shows the two I²C pull-up resistors (each 10 KOhm) as well as jumper J3, which is closed by default to allow Vcc (5V) provided to the bus.

Even if you do not use any I²C devices: as long the jumper is closed you may not be able to use D0 and D1 properly. Caution if you order our real time clock, I²C is needed for the communication with the clock.

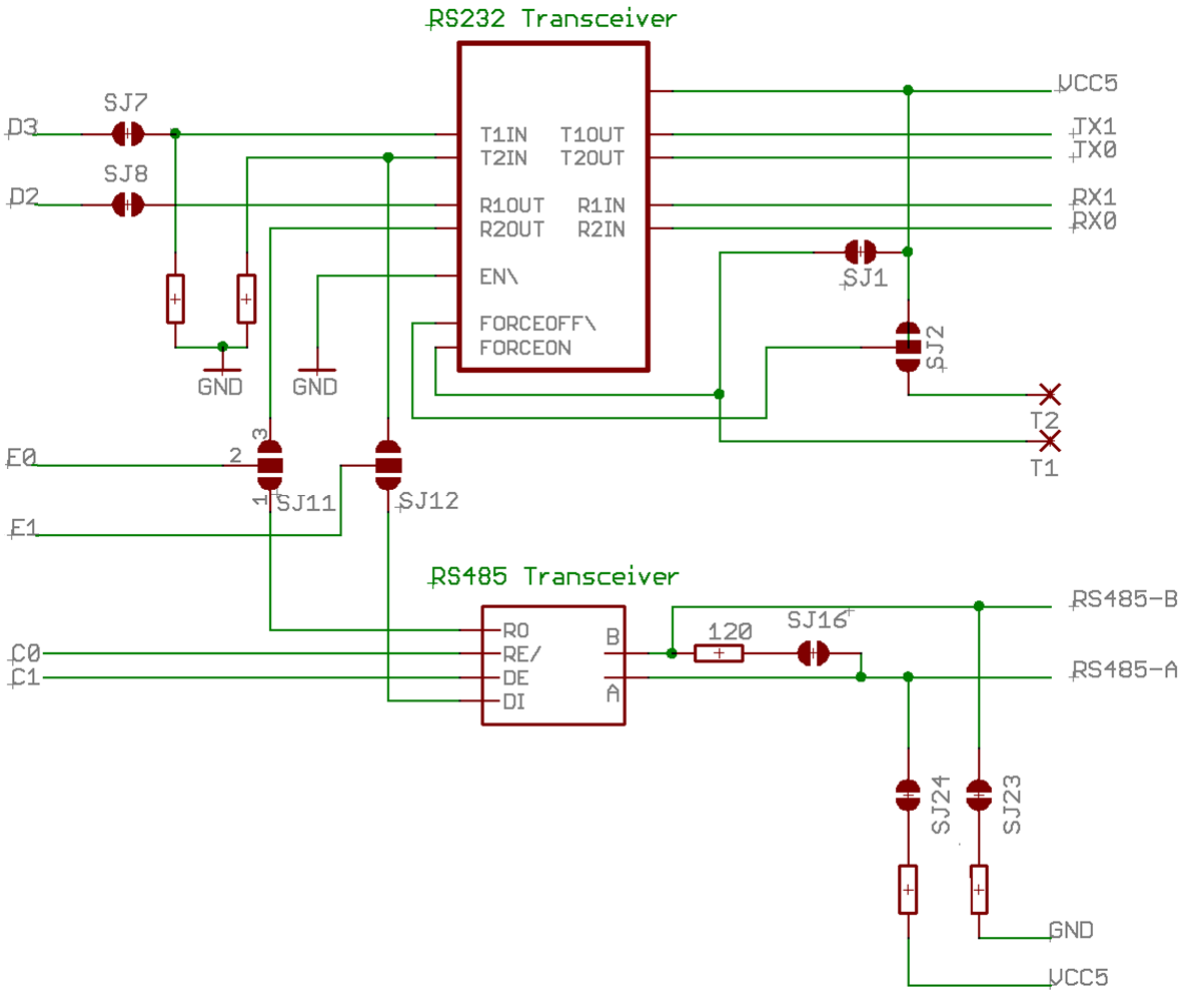
If you open the jumper (cut the thin wire between the two jumper fields at the dottet line) with a sharp knife, D0 and D1 are not connected to Vcc through the resistors anymore. With a drop of solder you may close the jumper again at any time.

Caution: If you open J3 the ports D0 and D1 are still connected to each other through 2 x 10 Kohm resistors. If this would interfere these ports, you need to desolder one of the two resistors aswell.

Serial Interfaces

The ATMega 128 / 2561 and the AT90CAN128 are offering two serial asynchronous interfaces: UART0 and UART1.

The connection of the RS232 and RS485 interface on our board happens as follows:

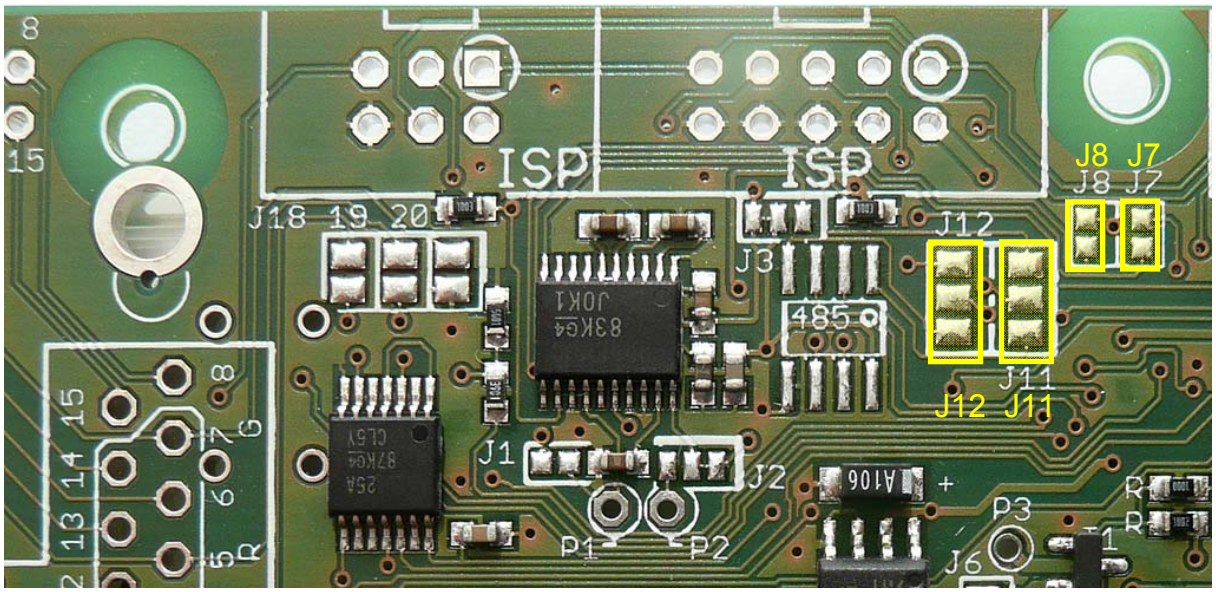


RS 232-interface

Important information upfront: If you want to use RS232 (UART 1) right away, you need to close the jumper J7 and J8. If you want to use the Dsub9 connector as a standard RS232 cable connection you need also to close jumper J18, J19, J20.

The pins of the serial interface 0 (UART0) of the ATmega microcontroller are also being used by the ISP programming interface. Connecting any transceiver by default to the ports E0 and E1 (UART0 pins of the controller) would result in a non working programming interface. Therefore you may close the jumper J11 and J12 if you do not need to program the board anymore. Or you may add a regular switch (at least at jumper J11) to allow you to open the connection to the transceiver which will prevent any ISP programming otherwise.

With UART1 the RS232 transceiver is also connected through two jumper which needed to be closed first. These jumper (J7 and J8) are open by default and need to be closed first.



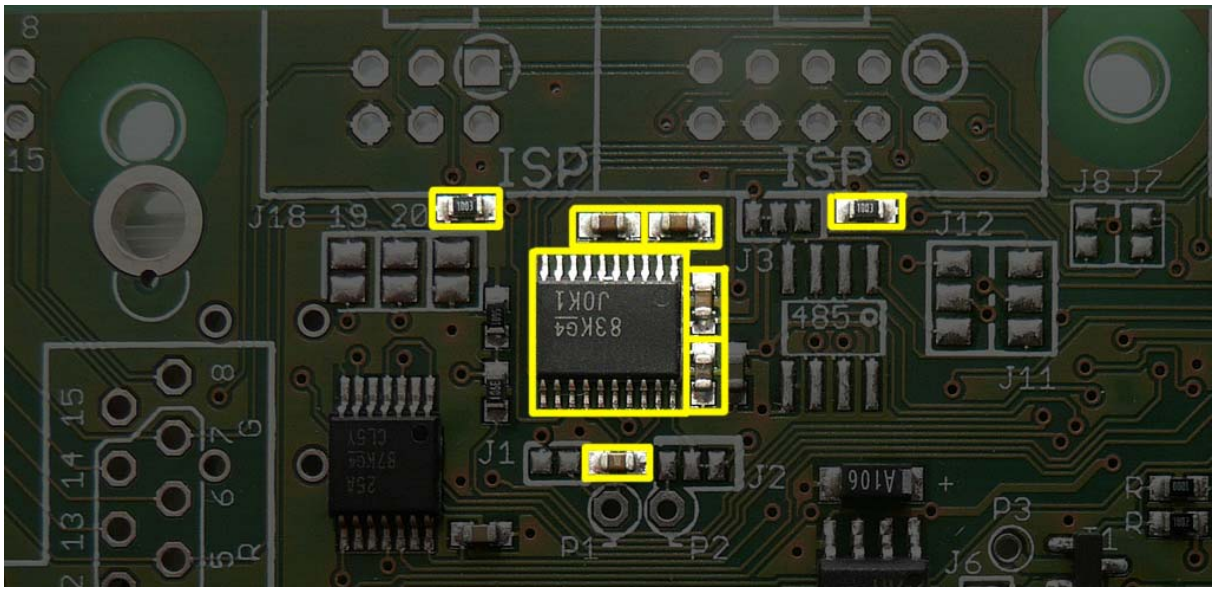
As UART0 can be used for RS232 as well as for RS485 you need to select the connection by configuring jumper J11 and J12. Thus you select, where the microcontroller ports E0 and E1 (UART1) will be connected to. By default all jumper are open. If you want to select RS232 you need to close the middle pad with the upper pad of the jumpers; if you want to select RS485 you need to connect the middle pad with the lower pad.

The serial interface 1 **(UART1)** is available with a high level signal at the pads Rx1 and Tx1 after closing the **J7 and J8**.

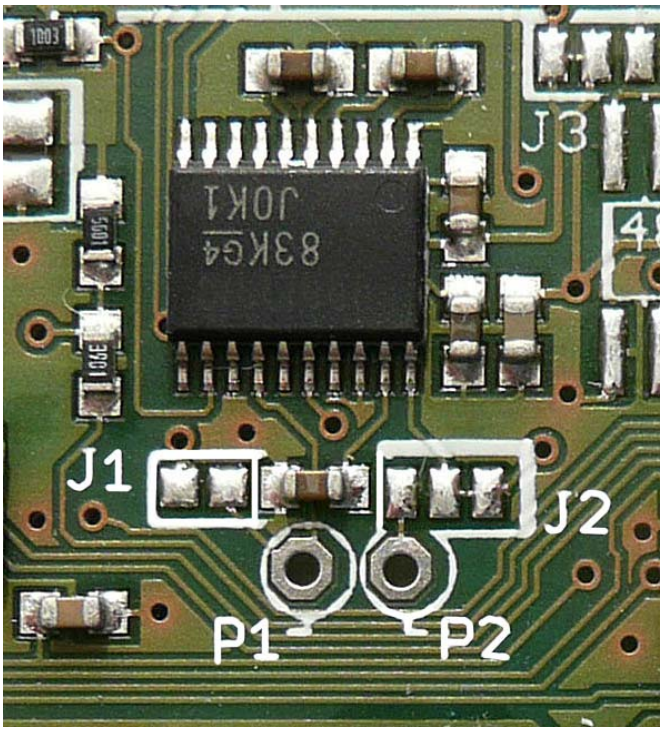
If you do need any high level signal but only to connect two controller modules, you may just do without using the high level transceivers. Just connect the controller ports and a common ground.

The D071x comes without a RS232 transceiver by default. This is an option you may order together with your module. The PCB then comes including the soldered parts. If you want to enhance your module later, you need the following parts (see exact part description on the schematic on your CD).

1 x SMD RS232 transceiver, TSSOP20
5 x SMD ceramic capacitors, size 603
2 x SMD resistors 100 KOhm, size 603



Below the RS232 transceiver you will find 2 jumper J1 and J2 as well as two solder pads marked with P1 and P2.



These pads and jumper were added to allow more detailed configuration of the different available shutdown mode of the transceiver.

The default state of the transceiver is auto-shutdown (energy save). If the transceiver is not detecting any other device it enters the shut down mode. Thus you will not be able to measure anything without having another RS232 device connected to the D071x.

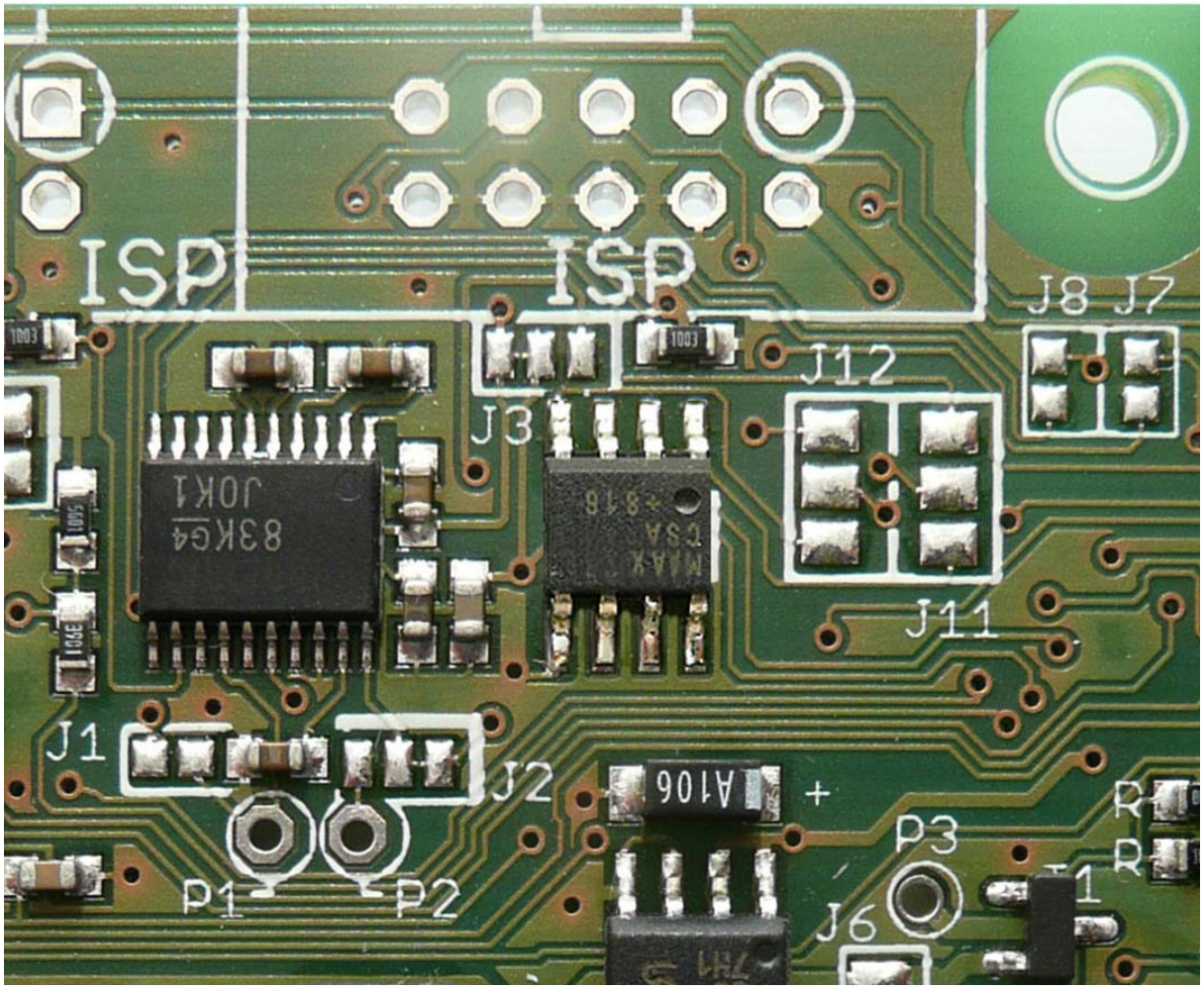
With the jumper J1 and J2 and the optional wire pads P1 and P2, you may configure the shutdown mode of the transceiver as you like. Please read the datasheet of the transceiver and our schematic for the different available mode.

<u>Again:</u> Without connecting another RS232 device the RS232 part of the D071x seems to be non functional due to the auto shutdown mode.

<u>Remark:</u> At page 66 et. seq. you will find more general details about RS232 and RS485.

RS-485 Interface

The D071x-Board by default comes without a RS485-transceiver. If you did not order the board with one soldered in you need to upgrade it by yourself. For this, you shall solder a Max485 chip to the shown position.



On the right of the RS485 transceiver, you will find two jumper (J11 und J12). What are they for? The ATMega uses for the RS232 or RS485 communication the same ports (here Rx0 and Tx0 – E.0 and E.1) – thus the serial interface 0 (UART0).

As you cannot connect RS232 and RS485 transceiver at the same time to the microcontroller, you need to select your decision here by configuring the jumper J11 and J12. Basically you select, where the ports E0 and E1 shall be connected to. By default, this jumper is open and not connected to any interface.

If you want to select RS232, you need to close the middle pad with the upper pad of the jumpers; for RS485, you need to connect the middle pad with the lower pad.

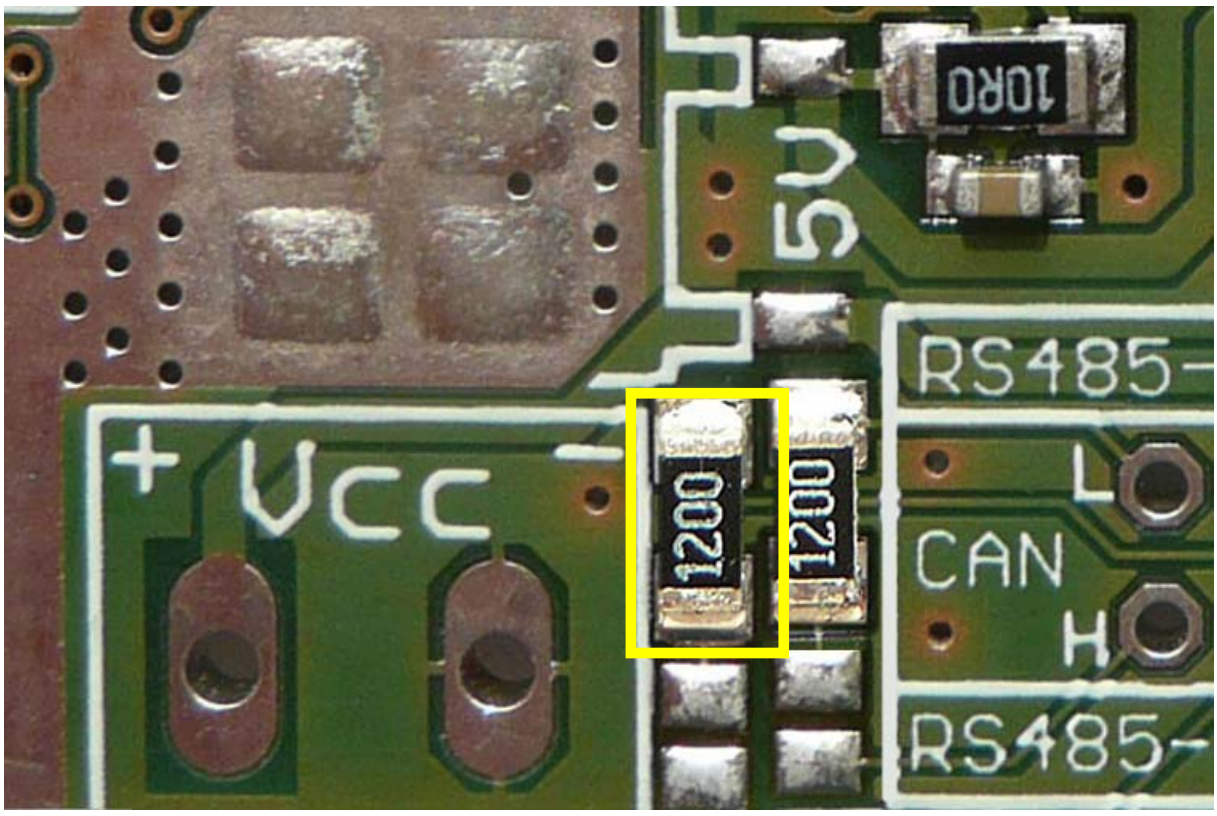
The RS485 transceiver is a halfduplex chip. To be able to do the correct switch between sending and receiving, you can separately switch the sending- and receiving-unit on or off.
RE (Receive enable) of the Max485 is connected to C.0 to the microcontroller;
DE (transfer enable) is connected to port C.1

RS485 usage with less than 5 Volt Vcc
If you want to use the D071x as a battery driven device with less than 4.5 Volts, the usual 5V type of the transceiver is not appropriate. You may select another type which works with 3-5 Volts but is more expensive.

Switch disconnection of the voltage supply for the RS485 Transceivers
At the upper left of the 485 transceiver chip, you will recognize another jumper J3. This allows changing the supply voltage from permanent Vcc (default) to a switched Vcc line. For changing this, you need to cut the connection between the middle and the left jumper field and connect the middle with the right field with a drop of solder.

RS485-Termination Resistor

At RS485 a termination of the bus might be needed at the last device of the bus. The board is prepared for this. If you order the board with RS485, it comes with the termination resistor (see photo below). By closing the soldering bridge (default is open), you may add the resistor to the bus as a termination resistor.



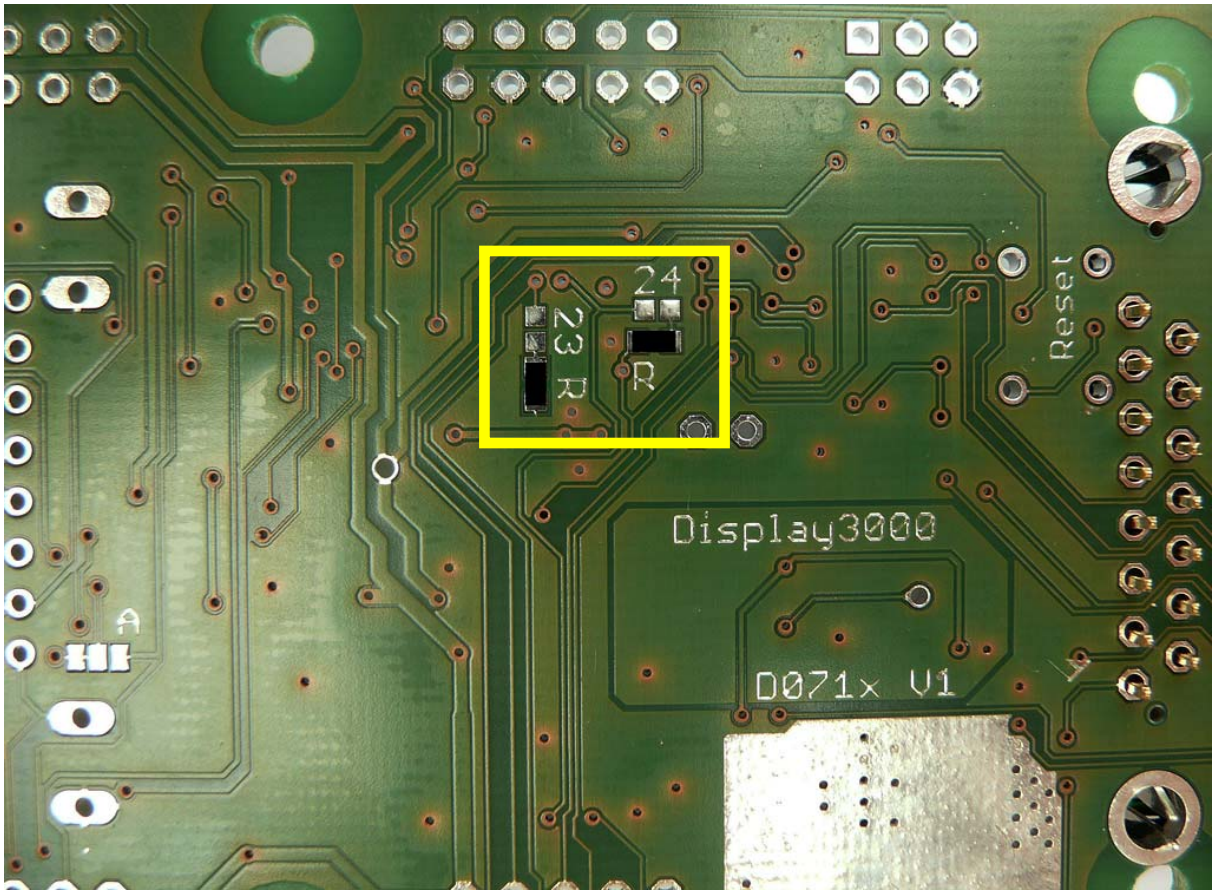
The photo shows two resistors. The left resistor is being used for the RS485 interface, the right resistor is being used by the CAN bus. The jumper (open by default) need to be closed to connect the resistor to the bus. If you did not order the RS485, the resistor is missing.

RS485 Pull-up / Pull-down

Perhaps (probably rarely) it might happen that the D071x shall act as the RS485 master board. Then it would be needed to setup a pull-up resistor on line A and a pull down resistor on line B.

The board is prepared for this too. For this, there are two hidden jumper and two hidden places for the resistors. They are hidden below the display. These resistors are not mounted by us and need to be added by you if needed. Closing jumper J24 is connecting the pull-down resistor and closing J24 is connecting the pull-up resistor.

You need to remove the display first to have access to these seldom used options.



Remark: At page 66 et. seq. you will find more general details about RS232 and RS485.

CAN bus interface

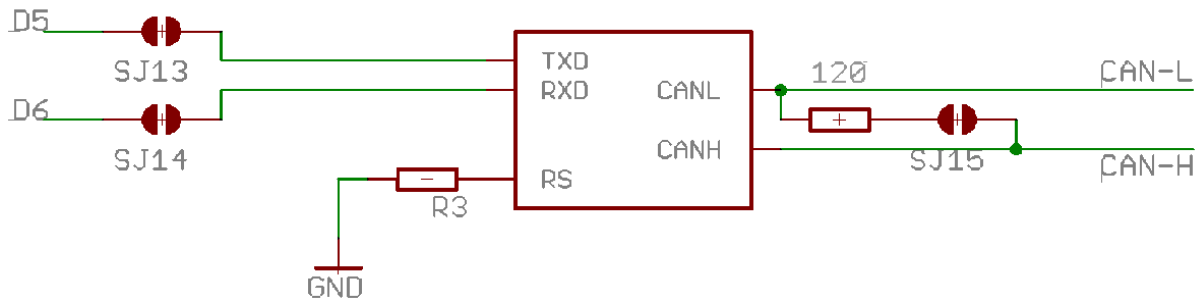
The CAN-Bus (Controller Area Network) is a so called field bus. It is a ansynchron, serial bus system.

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).



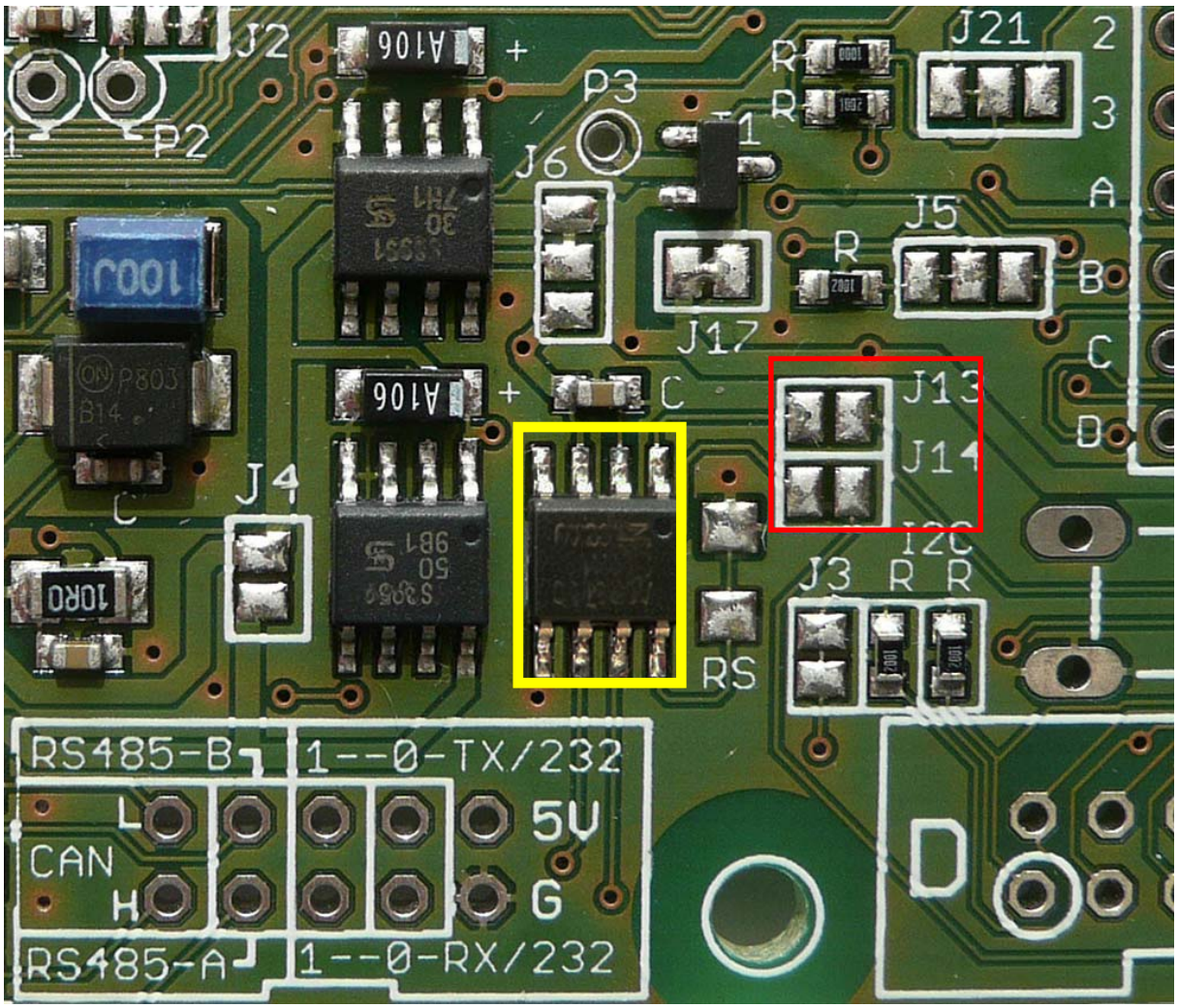
Information to 120 Ohm resistor: This is the bus terminator and should be 124 Ohm – usually 120 Ohms is used here and so did we.

R3 (resistor RS) NEEDS to be selected and soldered in by you. Please consult the data sheet and the *Application Note* of the PCA82C250 Tranceivers which we placed on our CD for selecting the correct value. As usually high speed is selected, the resistor needs to be 0 Ohm. Because of this, we did bridge the resistor by default. If you need a different value than 0 Ohm, you need to cut the thin connection inside the resistor pads first before you solder in a resistor.

<u>First hints:</u>

Highspeed:	0 Ohm – nothing to do for you – we already bridged the resistor pads.
1MBit/s	1 KOhm
100kBit/s:	47 KOhm

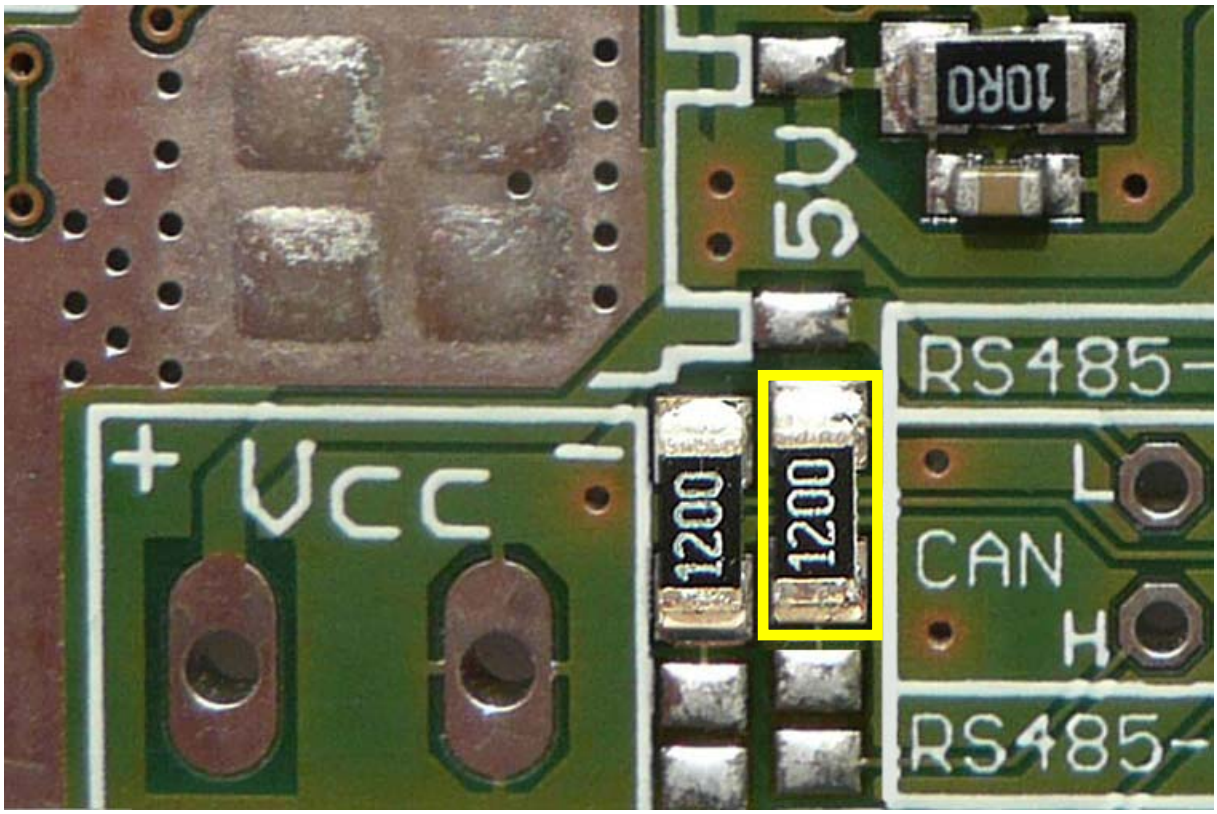
The CAN transceiver is been connected with port D5 and D6 of the microcontroller as these are the CAN ports of the integrated CAN controller. The two jumper bridges J13 and J14 are open by default. This means: the CAN transceiver is NOT connected to the micro-controller, if you receive the board. You need to close the two jumpers first.



The CAN transceiver is connected by default to the switched Vcc line. If you ordered the switch-off option and you switch off the power, the CAN transceiver is also switched off to save energy.

CAN-Termination Resistor

At CAN a termination of the bus might be needed at the last device of the bus. The board is prepared for this. If you order the board with CAN, it comes with the termination resistor (see photo below). By closing the soldering bridge (default is open) you may add the resistor to the bus as a termination resistor.



The photo shows two resistors. The left resistor is being used for the RS485 interface, the right resistor is being used by the CAN bus. The jumper (open by default) need to be closed to connect the resistor to the bus. If you did not order the CAN option, the resistor is missing.

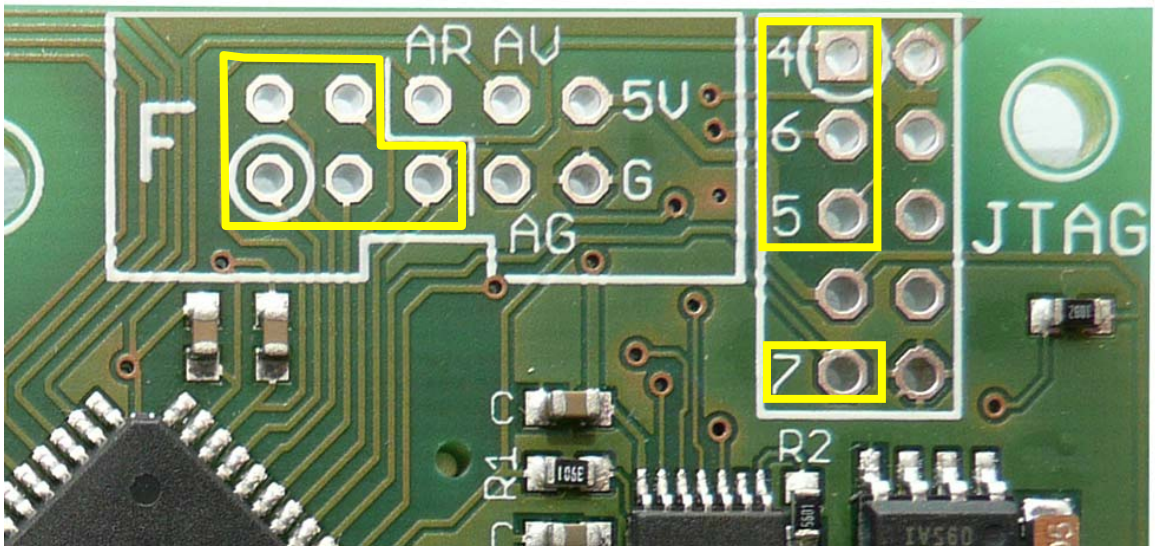
Analog Inputs (Analog-Digital-Coupler)

Some ports of the microcontroller can be used to read in analog values with a resolution of 10 bits (PortF0 to F7). 10 bits resolution means you may analyze changes of 0,5mV at a signal in the range of 0V to 5V.

For the most exact measurements, the analog-digital-coupler of the microcontroller can be connected with an external power source. The same applies for the reference voltage. These additional ports are AGnd, AVcc, ARef.

If the AD coupler does not need a very exact external supply, it is OK to connect AGND with GND and AVCC with VCC (the common power of the board). These pads are located at the connector for port F.

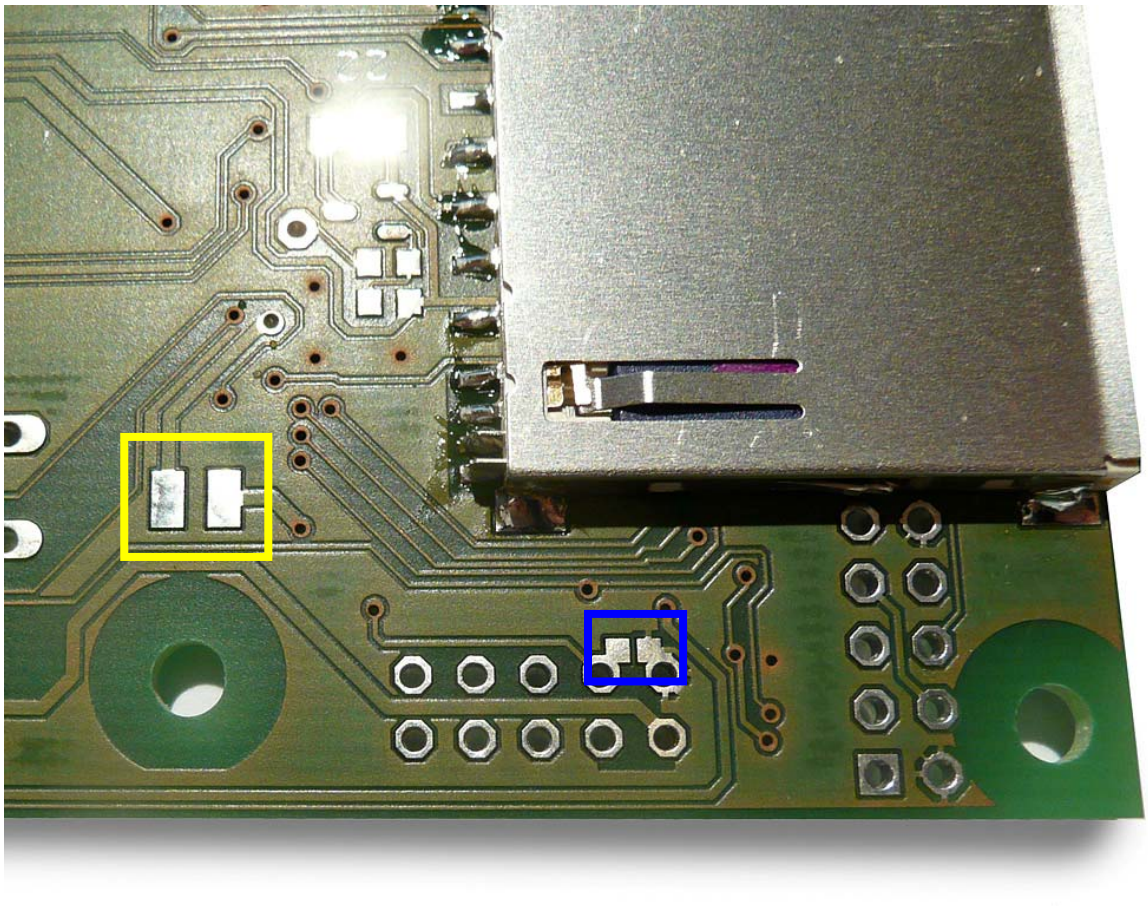
At the connector of port F only F0 to F4 are available. The ports F5 to F7 (and also F4) are available at the JTAG connector beneath. JTAG needs F4-F7, thus we removed these from the F-connector and placed ARef, AGN and AVCC instead (marked with AR, AB, AV on the board).



AVcc and AGND are not connected to 5V and GND by default.
If you want to measure analog signals, you need either provide the supply for the analog digital coupler or connect AGND with GND and AVCC with VCC.

For connecting this supply to the general supply we prepared two SMD pads on the upper side of the PCB (display side):

1. A soldering bridge for connecting AGND with GND is available at the F-connector (blue marking). Please close both pads with some solder to connect AGND to GND.
2. A 2nd bridge (yellow marking) allows the connection of AVCC to VCC (5V). As you see, this pad is quite larger as the AGND bridge. This pad allows also the soldering of a inductivity instead of just closing it. With a inductivity, you will set up a low-pass which will provide a more clean voltage to the coupler which allows a more exact measurement. As there is only a small current flowing, you may solder in a very small and flat inductivity (e.g. 10 µH, size 805; Farnell item No. 1669563). If you do not need the inductivity or if you do not have one handy, you may just bridge the two pads.



ARef is for providing the reference voltage of the internal analog digital coupler. For the measurement, the microcontroller may either use the internal reference voltage or an external voltage. ARef may then be connected with the general supply of the parts on the board (VCC = 5Volt). In this case, you may just connect ARef to AVcc or AVcc (disadvantage is a higher noise.

Caution: do not connect ARef to AVcc if you want to select the internal reference voltage of the microcontroller. The chosen reference voltage is available on the connector of ARef. As the internal reference voltage is lower than Vcc, this might then damage the AD coupler – at least your measurements will not OK.

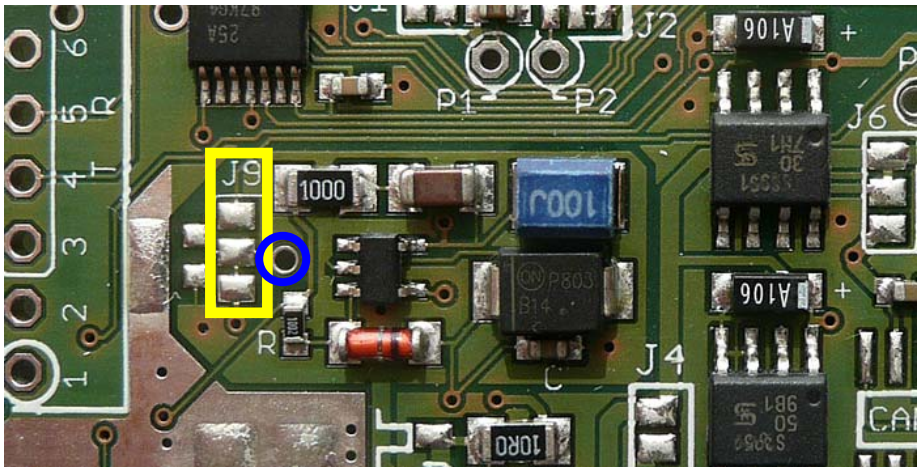
Switching the Display Lighting with the Microcontroller

As the lifespan of the display lighting is not indefinite and needs at least 50% of the energy consumption of the board, it makes sense to switch off or dim the lighting when the display is not needed. As we are using a transflective TFT the content can be read even without a lighting. A possible solution would be for example to dim the lighting to 50% after 1 minute of no input and to dim it even more (or switch off) after another few minutes of idle time. As soon as a switch has been pressed or if any alarm output is needed, the lighting can be switched to 100% again.

The booster on our board keeps track of production of the needed voltage by itself. It works from 2.5 up to 5 Volts input. By default the module has been set to continuous lighting.
You may change this by reconfiguring a jumper: Open the connection between the middle and lower pad of jumper J9 and close the connection between the middle and the upper pad.

Switching the Display Lighting

The lighting is driven by port B7. In your software, you need define port B7 as an output port. Whenever you set B7 to "1" the lighting is enabled – if Port B7 is defined to "0" the lighting goes off. Port B7 cannot be used then anymore for other usages of course. If you absolutely need B7 and you need to switch the lighting too, you can also do this: Open jumper J9 as described above. Do NOT close the jumper with the upper pad now. The wire pad beneath the jumper (blue circle) can now be connected by yourself to any available port with a piece of wire.



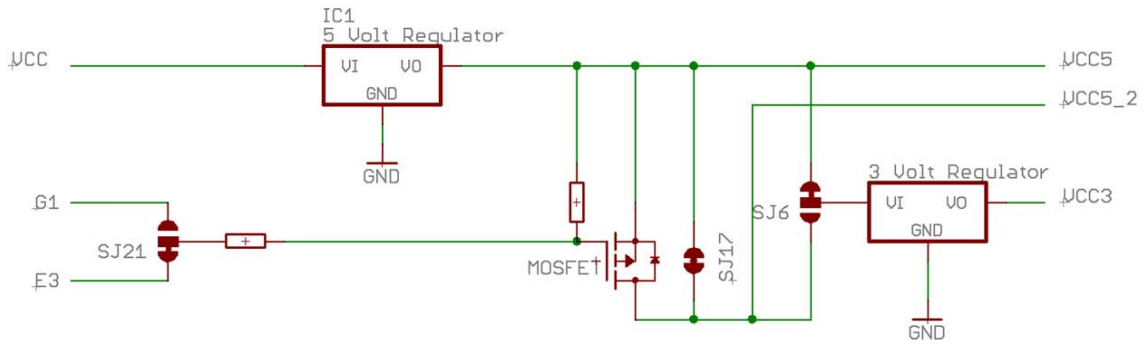
Please note: our program samples are defining PortB already as the display electronics also need Port B (SPI bus). If you also want to use B7 now, you might need to change our predefinitions.

Change in our Bascom code the line **Ddrb = &B01100110** to **Ddrb = &B11100110 .**
By this you define B7 as an output port now. With **Portb.7 = 1** or **Portb.7=0** you may now switch on or off the lighting.

Another interesting alternative for driving the display is the usage of pulsed signals. **By this you can also dim the lighting.** By using PWM, the microcontroller can provide such a signal without needing time or other resources. We are explaining PWM and how to use it in our appendix at page 72.

Switch-Off-Option

An option of the board is the possibility to switch-off unneeded electronic parts on the board by the microcontroller with a MOSFET to save energy.



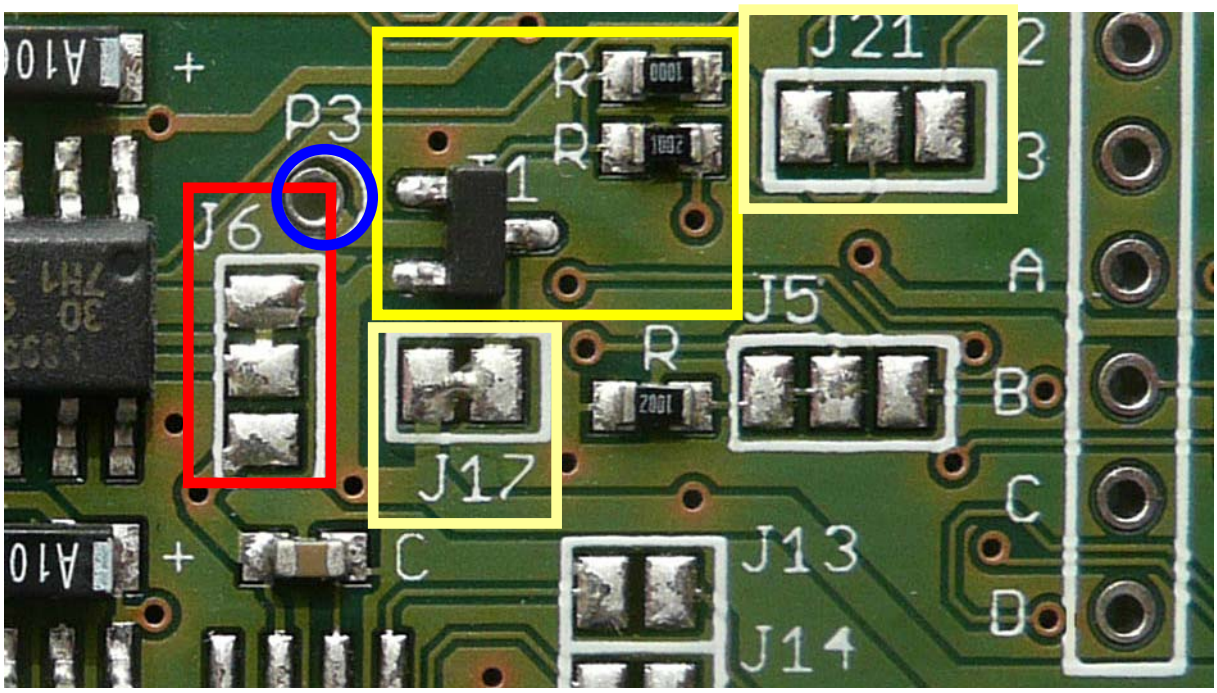
This MOSFET switches the main supply (called VCC5 in the schematics) to a second supply layer called VCC5_2.

The MOSFET can be switched by Port E3. By reconfiguring the jumper J21 (see photo on next page) you may change this port E3 to the alternative port G1. For switching the transistor, you need to define Port E3 (or G1) as output and set it to low ("0"). Low means switch devices on, high means: switch them off.

The areas you may switch off are:

- RS485 transceiver – you may select by J3 if the RS485 transceiver will be switchable or always switched on.
- CAN transceiver (always switchable)
- RS232 transceiver: not needed as it has an internal shutdown mode
- ~~5 Volts the SD-Card to the level transducer~~ (not possible – see also page 52 and 81)
- 3 Volts voltage regulator (and thus display, display levelshifter and SD card) - with J6 you select if the 3V regulator will be switchable or always on.
- Additionally, there is a wire pad you may use to switch any other external parts on and off (see blue marking on photo at next page).

Caution jumper J17: If you did not order the switch off option or if you do not need it, jumper J17 has been closed. The delivery (default) state of this jumper is "closed". If you want to use the switch-off-option, you must cut the jumper to open.



By default, the input of the 3 Volts voltage regulator is always connected to the 5 Volts supply and thus always working. If you want to switch also the 3 Volts part of the board, **you need to reconfigure jumper J6 (see red marking)**. You need to cut the connection between the upper and the middle pad and then, you need to close the connection between the lower and the middle pad of the jumper.

The 3 Volts regulator energizes the following modules:

- Display
- Display electronics
- SD Card
- 3 Volts pad at the Port B connector

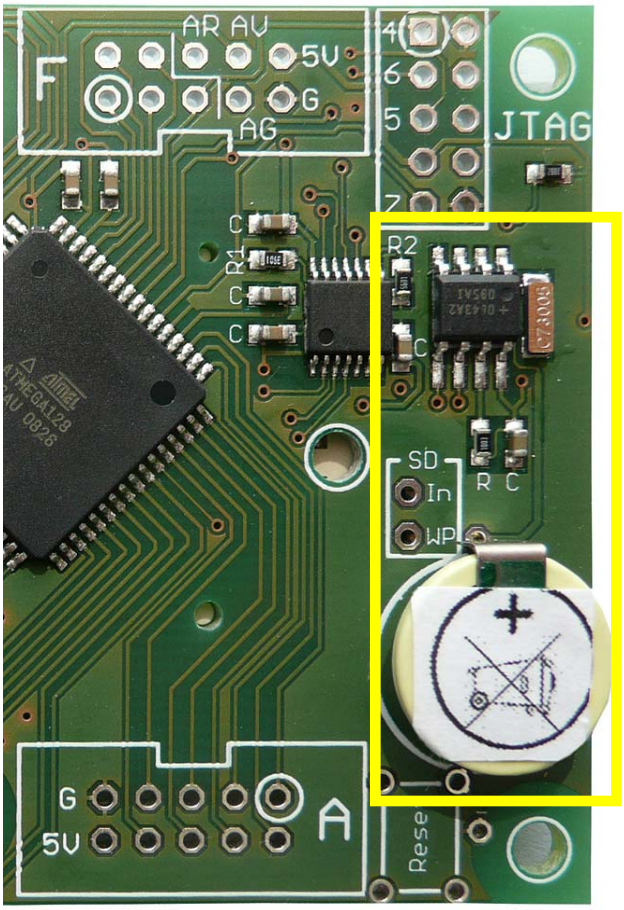
If you did switch off the 3 Volts supply, you need to reinitialize the applicable parts (e.g. display, SD card) as they were without power.

If you are using an SD card, you must take care that all open files are being closed before shutting down the power of the SD card. Otherwise, data loss and a damage of the FAT table of the card will be likely to happen – then, you will need to reformat the card.

Real Time Clock (RTC)

It makes a lot of sense to integrate a independent running real time clock to a microcontroller system as it allows the readout of the correct time and date even after a loss of power, a reset, a re-programming etc... Sometimes, this is not needed and bea bit of luxury, but sometimes, the date and time are quite important (e.g. when logging data on a SD card).

The D071x board offers such an option together with a miniature lithium battery (the yellow box shows the clock together with its clock crystal, the battery and some other needed parts).



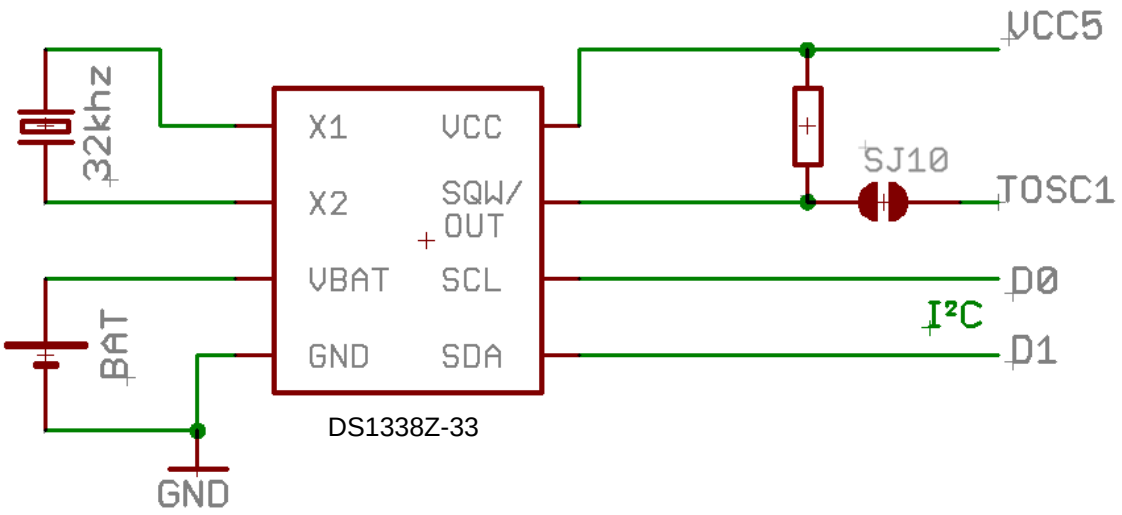
The battery overtakes the supply of the RTC when the main power fails even for a long time. The battery is able to provide power for approx. 2 years without power. As soon as the main power reocurs the battery shuts off. A lithium cell has an extremely small self discharge and can work in standby for more than 10 years.

The RTC chip that we are using is offering a very helpful additional feature: 56 bytes of user RAM which you can rewrite and read as often as you want (different to an Eeprom which allows only limited writing cycles). This RAM is also buffered and can be used to store configuration options or even run time data of the program. Thus a program would know what the status was when a reset happened and would even be able to continue a task.

The clock chip can be configured by software in a way that it provides a clock signal at the output pin RTC_Clock. This signal was routed by us to the TOSC1 port of the Microcontroller.

By closing jumper J10 (located directly at the microcontroller) you can connect this clock pad TOSC1 with the clock signal of the real time clock. This spares a separate clock crystal for the microcontroller as now the controller get the correct 32.768 Khz signal anyway. (Remark: the signal of the RTC can be programmed to different frequencies (e.g. 1 Hz).

Caution: The above paragraph does only apply to a ATMega2561 and AT90CAN128. According to the specs, the ATMega128 can not be provided with an external clock signal. Here are separate clock crystal between TOSC1 and TOSC2 is needed, no matter if you are using a RTC. Port G3 (TOSC2) is not usable anymore with a ATMega128 (see also page 31).



SD Card Module

With this enhancement, you may use SD cards as a low priced memory enhancement for your microcontroller. You even can very easily exchange data between the microcontroller and your PC by swapping the SD card.

IMPORTANT:

1) SD-Cards are not regular RAM, but flash memory. Each single memory cell has a life-span of about 100.000 write cycles (unlimited reading). It is said, that the card itself takes care about an even usage of all cells (this means, if you use only 10% of the capacity of the card, you may have 1 million write cycles available). Anyway: with a mistake in your software (a loop with permanent write access to the SD card) you may destroy your SD card in seconds. This means: SD cards are mainly usable as a storage place for reading (e.g. large graphics data) or for writing data you want to analyze later (e.g. a data logger which writes data every few seconds, where the data will later be analyzed by some spreadsheet software (e.g. Microsoft Excel). The SD card shall not be used for permanent ready/write cycles such as a RAM enhancement.
2) The software for the communication with the SD card itself was not developed by us. For questions regarding the software, please ask the author of it.
3) There are thousands of different SD cards of many manufactures out in the market. There might be some, which will not work with a microcontroller. Some are slow, even if they are marked as high speed. Some are sensible for timings and signal levels. We have found lot of pages on the Net where people had problems connecting SD cards to a microcontroller or having unreliable solutions. We wanted to offer a solution where these problems are taken away from users and we can tell you proudly: until today, not one single SD card was reported as non working or showing problems with our SD card module.
4) The usual software is only able to work with filenames of max. 8 characters length. So please use only max. 8 characters plus 3 for an extension. The file names "Logo1.bin" or „LogData1.dat" are fine where "Background.pic" or „LogDataSensor1.dat" are not.

Important Stuff to Avoid Data Loss:

If a reset, power loss etc... happens during writing something to a card or during the opening of a file, it will probably result in a damaged FAT file allocation table), a missing file, wrong file size etc... Because of this, you MUST close any open file before switching off the module.
To avoid this to happened, the following precaution is advisable:

If you write small amount of data unfrequent:
Gather your data in RAM first and then write the RAM buffer any xx minutes all at once or if you reached 512 bytes of data. For this you open the file, write the data and then you close the file immediately. The same applies for appending existing files with more data. By doing this, you will loose, in worst case, only the data you have gathered between the last writing session – if the power loss does not happen exactly at the writing moment (which you could bypass e.g. by working with two parallel files where you write to consecutively).

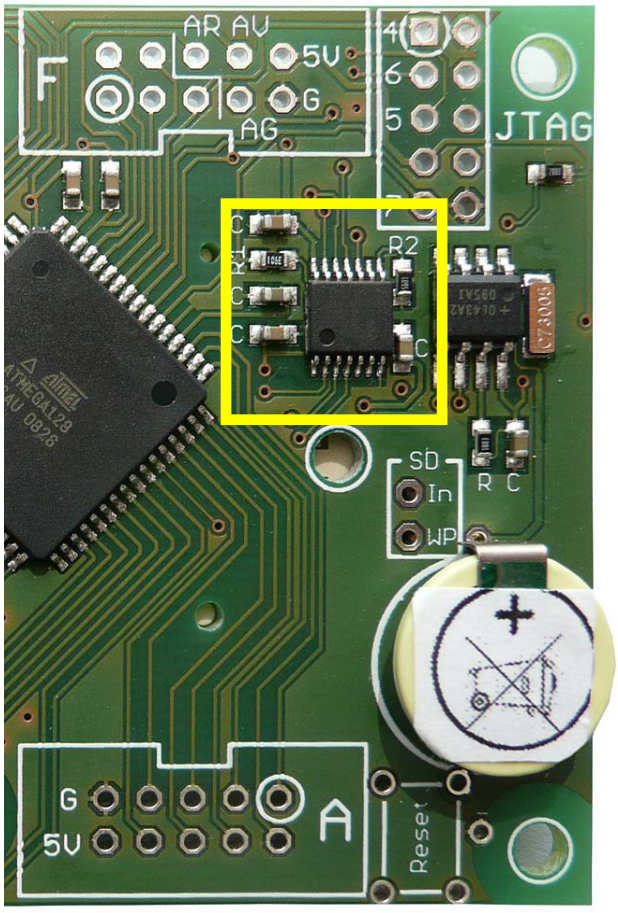
Writing large data amounts:
Usually you will not do this permanentely as this will reach quickly the 100,000 write cycles boundary. Because of this, we would like to advise to do this: Open file – Save/append data – and Close file.

The SD card enhancement accomplishes several tasks:

- Plug for the SD card
- Bidirectional level shifter from 3,3-5 Volt (microcontroller) to 3 Volt (SD card) and vice versa
- Decoupling the complete SD card module from the SPI bus with its tri-state outputs

Why is a Bi-directional Levelshifter with Tristate Outputs needed?

The SD card needs a supply of 2.7 up to 3.6 Volts and the needed current varies between a few mA up to several hundred mA (for a short time). Following this, a regulation by a real voltage regulator down to 3 Volts is obligatory. The D071x is equipped with it.



A direct connection between the microcontroller (driven with 5 Volts) and the SD card (3 Volts) would destroy the card as the signal level would be too high. The level of the signals need to be transformed from 5 Volts to 3 Volts (and vice versa).

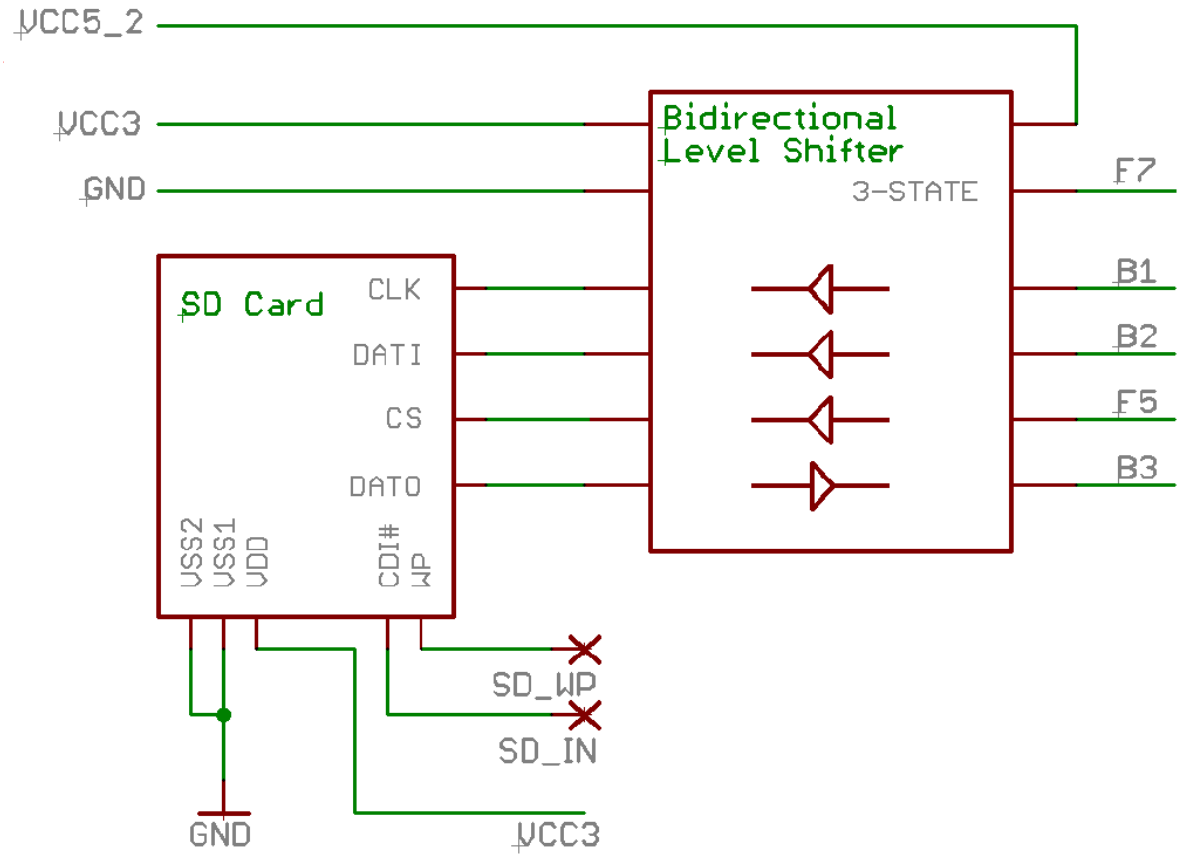
Additionally, the SD card would block the SPI bus with a steady high signal (even if unused), we have to provide the possibility to remove the SD card from the bus.

Therefore, we equipped the D071x with a bi-directionional levelshifter (this means it transforms from 5 Volts to 3 Volts and from 3 Volts to 5 Volts) which can be removed completely from the SPI bus by desactivating it (the yellow mark shows the level shifter with some more needed parts).

Tristate-Outputs

As explained above, the output of the level shifter will need an option to be able to remove it from the SPI bus as they would otherwise block the bus (and thus, the function of the display). This happens with the activate line which is run by Port F7. There need to be a high signal on F7 to be able to communicate with the SD card. After communicating a low on F7 will free the SPI bus for other devices (e.g. for the display).

Schematics of the SD Card Module



The SD card module needs, beside the three common SPI ports Clock, MOSI and MISO (B1, B2, B3), two more ports of the microcontroller: F5 which is the CS line and F7 which is used for activating/deactivating the level shifter (as explained above).

Errata:
Unfortunately, the level shifter will be a problem if the voltage supply is been switched off but the microcontroller not. Because of this a minor correction is needed on the PCB.
The supply pin of the level shifter needs to be removed from the switched VCC5_2 line and to be connected to the continuous supply line VCC5.
This will not create problems with the energy consumption as the level shifter only needs 1µA in the tristate-mode.

If you received your board after the 15 August 2009 and ordered the SD card option as well as the switch-off option, we already made this correction on the PCB.
If you need to do it by yourself, you will find instructions on page 81.

Conflicts at the Configuration of the SPI Bus

Different SPI devices will share the same bus but may expect different data formats. In this case with the color display and the SD card using the bus this happens as well:

TFT: Polarity = Low, Phase = 0
SD Card: Polarity = High, Phase = 1

Also the SD card needs a slower data transfer than the display.

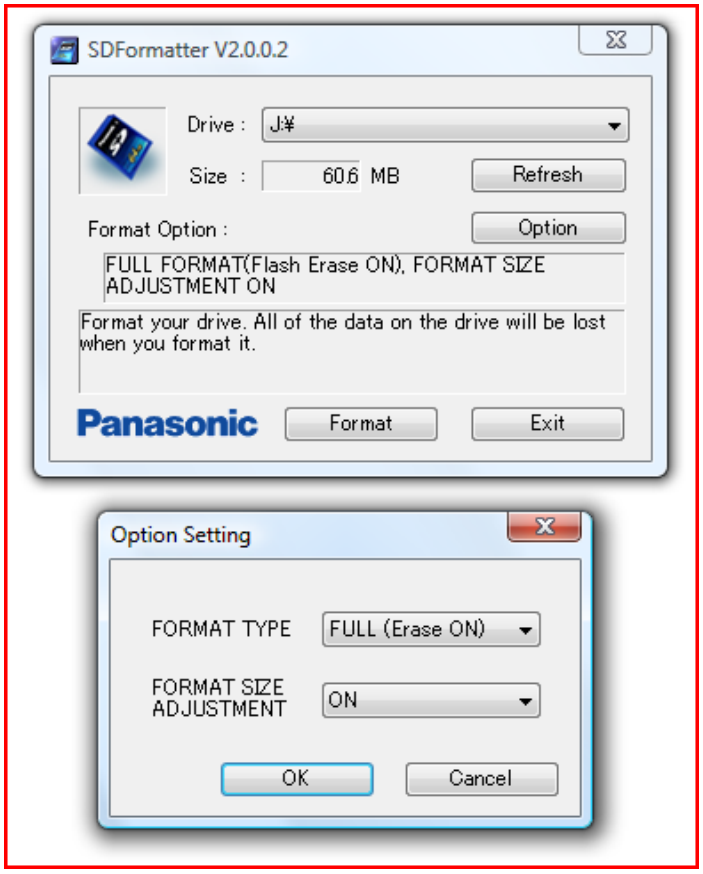
This means you need not only switch the activate line (Port F7) of the card and CS line, you also need to reconfigure the SPI bus to the new data format!

In our software example, we already created a subroutine for this. These routines are called *Activate_Display* and *Activate_SD* which you shall call if you change the communication from one device to another.

Formatting of a SD Card

If you are able to use your SD card correctly right from the beginning, you do not need to care about any formatting. Just skip this chapter then.
If formatting through MS Windows is not working, you might need to use a separate tool for formatting your card.

At the following link: http://panasonic.co.jp/pavc/global/cs/sd/download/sd_formatter.html you will be able to download the software SD Formatter from Panasonic, with which you will format your SD card first. Due to copyright reasons, we can only provide this link.



The manual for this software is available here:
http://panasonic.jp/support/global/cs/sd/download/ftp/manual2007e.pdf

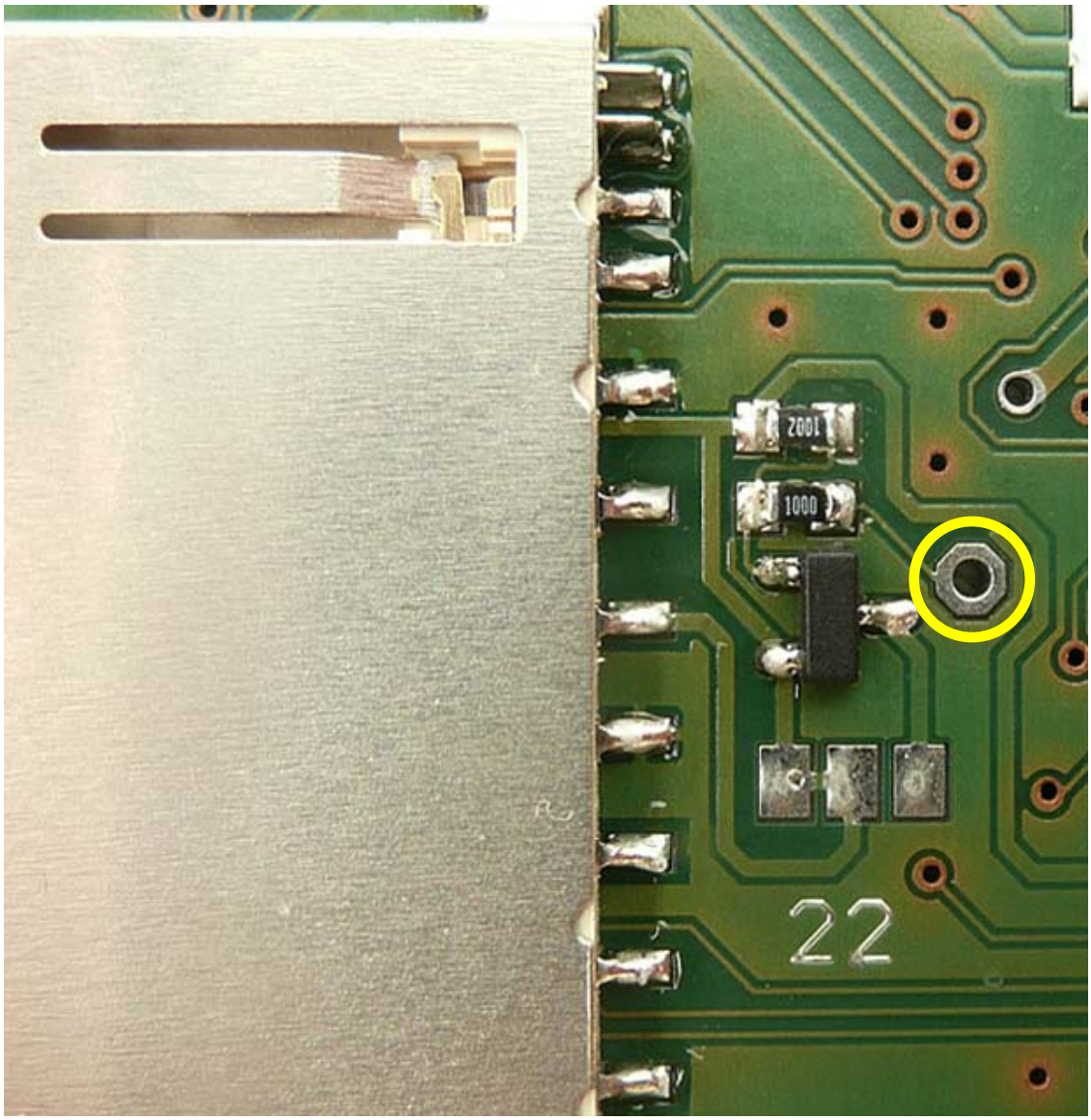
If the upper links are not functional one day, you may find a backup of software and manual here: http://www.sdcard.org/about/downloads/

Hint 1: Often the default settings are fine:
1) Quickformat
2) Format size adjustment: off

tal camera (most if no

Hint 2: We found SD cards which could be accessed by any possible devices, but not the microcontroller. If all trials formatting them from Windows or the Panasonic formatting tool failed. A solution is the format option of a digital camera (most if not all will have such an option hidden somewhere in the expert menu).

Switch-off the supply of the SD Card



In case that you are using a battery driven device and you want to switch off only the SD card (which is consuming a lot of current), we prepared the board for this: On the upper side of the PCB (display side) you may add a P-channel MOSFET and two resistors to be able to switch the SD card on and off. Jumper J22 is responsible for the selection. By default, the SD card has continuous supply. To be able to switch it off, you need to reconfigure the jumper. The wirepad allows you to connect the switching transistor to any available port you like by a piece of wire. The MOSFET need to be variant with a V_{GS} of 3 Volt or less in a SOT23 package.

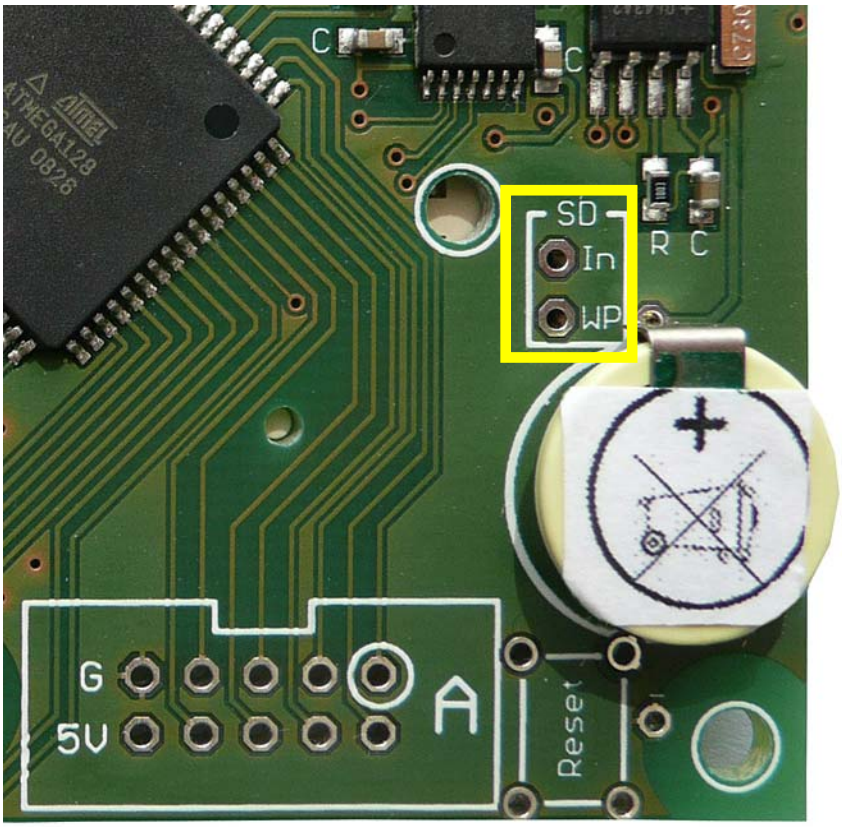
Recognition of Write Protection / Card Removal

Sometimes, it might be necessary to recognize if a card is either write protected or even physically available (or being removed accidently). This is not possible by software only.

The write protection switch at the SD card is not been recognized by the card itself – it is just a piece of plastic. You may write to the card even with an enabled write protection switch.

The SD card holder offers two mechanical switches for card insertion and write protect recognition. These two switches were routed by us to two wirepads marked with IN (insert) and WP (write protect).

These two wire pads are not routed to any microcontroller port. You need to solder a wire to any available port. The contact of the card module will pull the IN line down to GND if a card is present and pulls the WP line to GND if the card is NOT protected.



CAUTION: The ports you connect with the WP and IN line shall NOT be set to "Output".

Only a setting as an "Input" with a switched on pull-up resistor is allowable.

If you set these ports as output, they will be drawn against GND by the mechanical switches of the SD card module and thus short circuited. The output transistors of the port will burn through immediately and the port is not usable as an output port for you anymore.

Using the Housing BS600 / BS601

We tailored the D071x to the hand held enclosure. This means that it fits exactly to the dimensions and the mounting bolts.

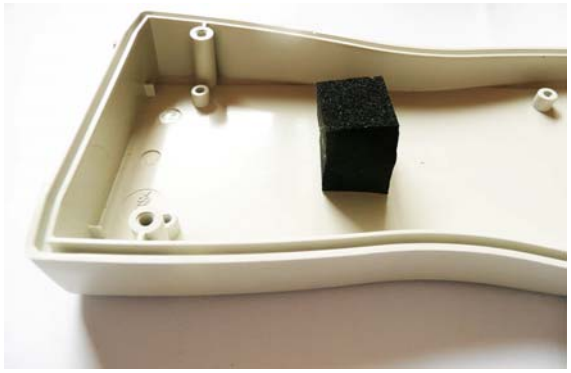
Some aspects need to be noticed though:

a) If you ordered the SD card module, the lower middle mounting bolt needs to be removed as the SD card holder is placed there. Just cut it away with a wire cutter (see photo). This is no problem as the PCB will be fixed by two more screws at the lower end – having three mounting screws would be luxury.

b) At the upper end, the housing does not offer mounting bolts in the upper part of the housing. The PCB is well fixed with the screws at the lower end but at housings without battery compartment they should be fixed somehow for security reasons nevertheless. There are a few possible ideas (beside glueing):

   a. You might screw two stay bolts to the lower shell (see photo) which will then fix the PCB when you close the housing or

   b. you place some foam rubber at the lower shell which will then press the PCB against the upper shell when you close the housing (see photo)



These tasks are not need when using a housing with battery compartment as the PCB will be fixed by the compartment in this case.

c) If you want to use the membrane keypad, you need to mil an opening for the FPC cable first. This could be done easily by drilling a few 2-5mm holes side by side which you may then enhance with a knife or a file to a slot. You do not need to be very exact here as this area will be completely covered by the membrane keypad and thus will be invisible and water proof.

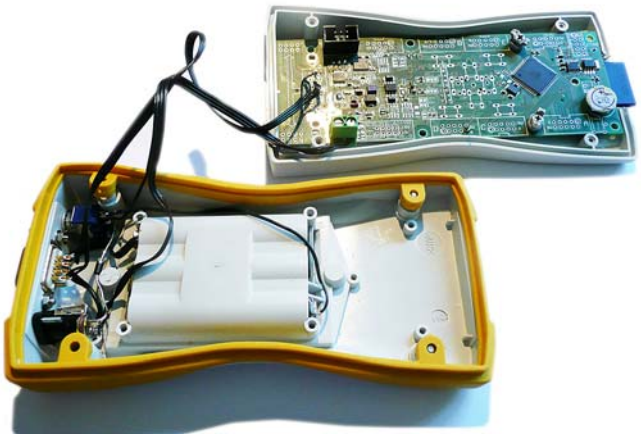
d) When using the membrane keypad it will make sense to fold the cable twice (Z-fold) as it will be easier to mount. We do suggest two folding areas (see photo): fold the cable (measured from the end of the cable contacts) at position 3cm and 4cm. (equals 1.18" and 1.57"). If possible use the FPC connector we are providing. The usual available FPC connectors are too thick (too high) to fit between PCB and housing shell.

e) For the SD card or a DSub connector you need to mill or to saw a notch into the housing. We do recommend:

a. For the SD card it will be OK to rasp a few mm of the housing edge as shown at the photo. This can be done quickly and will look nice.

b. For the DSub connector, we do recommend a connector that you can mount from the outside to the housing by a nutbolt (e.g. Farnell item No. 1217993). See photo alongside (here showing a connector together with a switch and a power socket).
Cutting the opening is not critical as it will be covered by the connector bezel later. Additionally, the connector will get a strong mechanical support by mounting it to the outside.
The photo alongside shows the connector mounted in the lower shell of the compartment – it is also possible to mount it in the upper shell.
The housing still has enough space for a small addon PCB for specified tasks which could be: charging circuit, USB circuit, optocoupler, DC booster, wireless modules etc.
If you like, we can plan, realize and produce such a add on PCB for you.

c. If needed, you just use short wires to connect the PCB with any additional connectors or add on circuits as shown at the photo. Of course, you will be able to use the DSub connector as shown on page 19. You just have to mill a larger cut out into the housing to be able to connect with the mating connector.



Overview Jumper / Soldering Bridges

On the preceding pages, you discovered many configuration options which often uses jumper or soldering brigdes. Following is a table with all jumpers and their function. The default position is printed in bold.

Jumper	Positioning on PCB	See page	Explanation
TJ_A	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between **G0 (default)** and D2
TJ_B	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between G2 and **G3 (default)**
TJ_C	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between **D5 (default)** and G2
TJ_D	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between **G1 (default)** and E7
TJ_E	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between **E7 (default)** and D6
TJ_F	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between E7 and **D6 (default)**
TJ_G	Display side, at switches	**Error! Text mark not define.**	Chooses switch allocation between G1 and **E5 (default)**
J5	Part side, middle of PCB		Level shifter display. The jumper switches between **always on (default)** and switchable by port F6.
J6	Part side beneth the 3 Volts voltage regulator	**Error! Text mark not define.**	Chooses if the 3 Volts regulator may be switched off by switch off option. **Default: always on**
J7	Part side between PortB and ISP	**Error! Text mark not define.** **Error! Reference could not be found.**	Connects Port D2 with TIn of the RS232 transceiver. **Default: open**
J8	Part side between PortB and ISP	**Error! Text mark not define.**	Connects Port D3 with ROut of the RS232 transceiver. **Default: open**

J9	Part side, left (booster area)	**Error! Text mark not define.**	This jumper configures if the display lighting **is always active (default)** or being switchable by the micro-controller (Port B7)
J10	Part side at the microcontroller upper left side	**Error! Text mark not define.**	Connects TOSC1 with the Clock Output of the real time clock. Only usable if no separate clock crystal is being used and if the optional real time clock is used. **Default is open** (though we might close it during production if you order a RTC and no clock crystal)
J11	Part side, below the 10pin ISP connector.	**Error! Text mark not define.**	Sets if port E1 shall be connected with the RS232 or the RS485 transceiver. **Default: Open (not to connected to any transceiver)**
J12	Part side, below the 10pin ISP connector.	**Error! Text mark not define.**	Sets if port E0 shall be connected with the RS232 or the RS485 transceiver. **Default: Open (not to connected to any transceiver)**
J13	Part side, right of the CAN transceiver	**Error! Text mark not define.**	Connects CAN-TxD with port D5, **Default: open.** Needs to be closed for using CAN
J14	Part side, right of the CAN transceiver	**Error! Text mark not define.**	Connects CAN-RxD with port D6, **Default: open.** Needs to be closed for using CAN
J15	Part side, between VCC supply connector and serial interface connector area	**Error! Text mark not define.**	Connects the 120 Ohm termination resistor to the CAN bus. **Default: open**
J16	Part side, between VCC supply connector and serial	**Error! Text mark not define.**	Connects the 120 Ohm termination resistor to the RS485 bus. **Default: open**

	interface connector area		
J17	Part side, center, below the MOSFET	**Error! Text mark not define.**	Bridges the optional MOSFET fort he switch-off option. **Default: closed**
J18	Part side, upper left area	**Error! Text mark not define.**	Connects Rx with the Dsub connector **Default: open**
J19	Part side, upper left area	**Error! Text mark not define.**	Connects Tx with the Dsub connector **Default: open**
J20	Part side, upper left area	**Error! Text mark not define.**	Connects GND with the Dsub connector **Default: open**
J21	Part side, centered	**Error! Text mark not define.**	Decides, if the MOSFET of the switch-off option is being switched by port G1 or E3. **Default is port E3**

Overview of the Ports allocation

Please, also have a look at the separate Excel-Spreadsheet file called: *Ports usage DS071x.XLS*

Technical Data Display Module Kit:

Item D071x:

Size:	146 x 54 mm (5,75“ x 2,12“ approx. 8 mm height (0,31”) (including display, no switches)
Weight:	approx. 35-40 gram (depending on options)
Voltage Supply: (see also chapter about needed voltaged)	3 to 12 Volt DC; optional up to 18 Volt
Microcontroller:	Atmel ATMega 128 (Option: ATMega2561, AT90CAN128 *1 *2) 128 KByte program memory (256 KByte *2) 4 KByte RAM (8 KByte *2) 2 KByte Eeprom max. 16 MHz clock (depends on ordered option)
Display:	176 x 132 pixel, 65.536 colors Active diagonal area 2,1“ (53 mm)
Background lighting:	3 white LED, Color Ranking 5,6
Dot Pitch H & V:	0,237 mm (0,1“)
Viewing angle:	6 o’clock (controller: 12 o’clock)
Lifespan working:	minimum 5 years
Lifespan display lighting at 100%:	minimum 5000 hours (=brightness went down to 70%)
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)

Optional the board may be equipped with:
*1) ATMega2561 (256 KByte Flash, 8 KByte RAM, 8 KByte Eeprom)
*2) AT90CAN128 (like Atmega128 but with CAN-interface)

The microcontroller has been pre configured by us:

Fusebit: 0100 (8 MHz intern) or 1111 if you ordered an external crystal
Fusebit: Preserve EEPROM when chip erase
Fusebit P: ATMega 128 Modus (at a ATMega128 controller)
Fusebit: **Brown Out** to 2,7 Volt

Eeprom and program memory were preprogrammed for testing the module

Frequently Asked Questions:

I am not able to use Ports F4 to F7

Ports F4 to F7 are not available as long as the JTAG fuse is switched on. If you do not need JTAG you may switch off the fuse. Otherwise you may switch off JTAG by software temporarily in your software.

I am not able to use port D0 and D1

These ports can only be used if you do not need I²C (TWI). We do need these ports for the integrated real time clock. If you do not order the RTC, you may open jumper J3 and then use D0 and D1.

I am not able to use the ports D5 and D6

Probably you ordered a AT90CAN128 microcontroller and you closed the jumper J13 and J14. D5 and D6 are the data lines for the CAN transceiver and cannot be used together with the CAN module. If you do not need CAN open jumper J13 and J14 (default: open).

I cannot communicate with the real time clock

Please check if jumper J3 is closed (default: closed). This jumper connects the I²C-pull-up-resistors with the main supply

I cannot communicate with the SD card

Please check if JTAG has been switched off. The SD card needs port F5 and F7 and these are only available if JTAG has been switched off. Hint: JTAG can be switched off by software temporarily (see specs sheet of the microcontroller)

I cannot do any analog measurements

Probably you did not connect AVcc, AGND and ARef. Please read the chapter analog measurements on page 43.
If the analog inputs 0 to 3 are working but not 4 to 7, please read the section "*I am not able to use Ports F4 to F7*" above.

I connected a RS232 cable to the DSub9 connector but I am not getting a connection

1) Are jumper J18, J19, J20 closed?
2) Are you using UART1 (Rx1/Tx1) in your software?
3) Do you use a standard 1:1 cable?
4) Are the jumper J7 / J8 closed ?
5) The RS232 transceiver goes into shutdown if there is no other device detected. There must be a signal on Rx to wake up the transceiver.

I cannot use CAN

1) Are jumper D5 and D6 closed?
2) Did you add the termination resistor (if needed)?

The board works fine but after a while the lighting starts to blink and the program crashes or the board resets by itself.

Probably the voltage regulator gets too hot and enters the emergency shutdown mode. Please read the chapter voltage regulator on page 11. Please take in mind that by adding more load to the board (e.g. programming interface, external circuits etc.), the calculated limits will not be valid anymore. Reduce the input supply or use a larger voltage regulator.

Everything works fine but sometimes the display goes crazy

The glimmering area at the lower area of the display contains the display controller. This controller is light sensitive. Either build the PCB inside a housing which covers this area or place some dark tape over this area.

Suddenly I cannot reprogram the controller – no access anymore

This might have several causes:
1) The microcontroller became damaged by electro-static discharge (ESD). If you are statically charged and you touch the PCB there might be a potential of several thousand volts flowing through the board. The current may be minimal and you even might not feel anything, but the controller or other devices become destroyed. Even if it worked out several hundred times, one day anybody could face this problem.
A good strategy is to touch at least a grounded part before touching some electronics. This might be a GND at your oscilloscope, at the shielding at a USB connector, the pipe of a radiator, etc... It is usually ok to do it onces after sitting down as you are charged up by running around, not by sitting on your desk.
2) A wrong programmed fuse: Do not play around with the fuses with none or only superficial knowledge. With a wrong fuse setting, you can easily disable the microcontroller. We would then have to swap the controller on your expense. If you are not sure about a fuse setting, please write a mail or call us.

Single ports are not working as expected

Either you are using a port which has an alternative usage (check our Excel sheet with the port usage table) or a wrong usage (now or some time before) did damage a port (e.g. short circuit).
If the later applies, the port is no more usable. Hint: Sometimes, if a port output is burned, it can still be used as an input port. Give it a try.

I want to use the real time counter but it is not working and also the appropriate time is not working

1) If you own the real time clock: is jumper J10 closed?
2) If yes: did you program the real time clock so it is emitting a clock signal (check the SQW/Out pin with an oscilloscope)? You shall measure there (or directly at the jumper J10) a 32.768 Khz signal.
3) If you do not own a real time clock: Is there a clock crystal together with 2 x capacitors mounted? (see page 31)? Jumper J10 shall not be closed then!

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

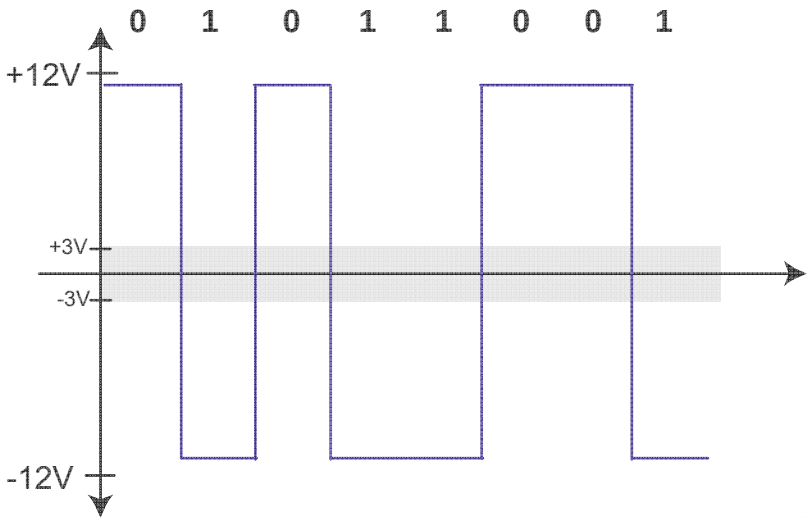
Author of this manual: Peter Kuesters

Appendix: RS232 and RS485 addl. Information

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.

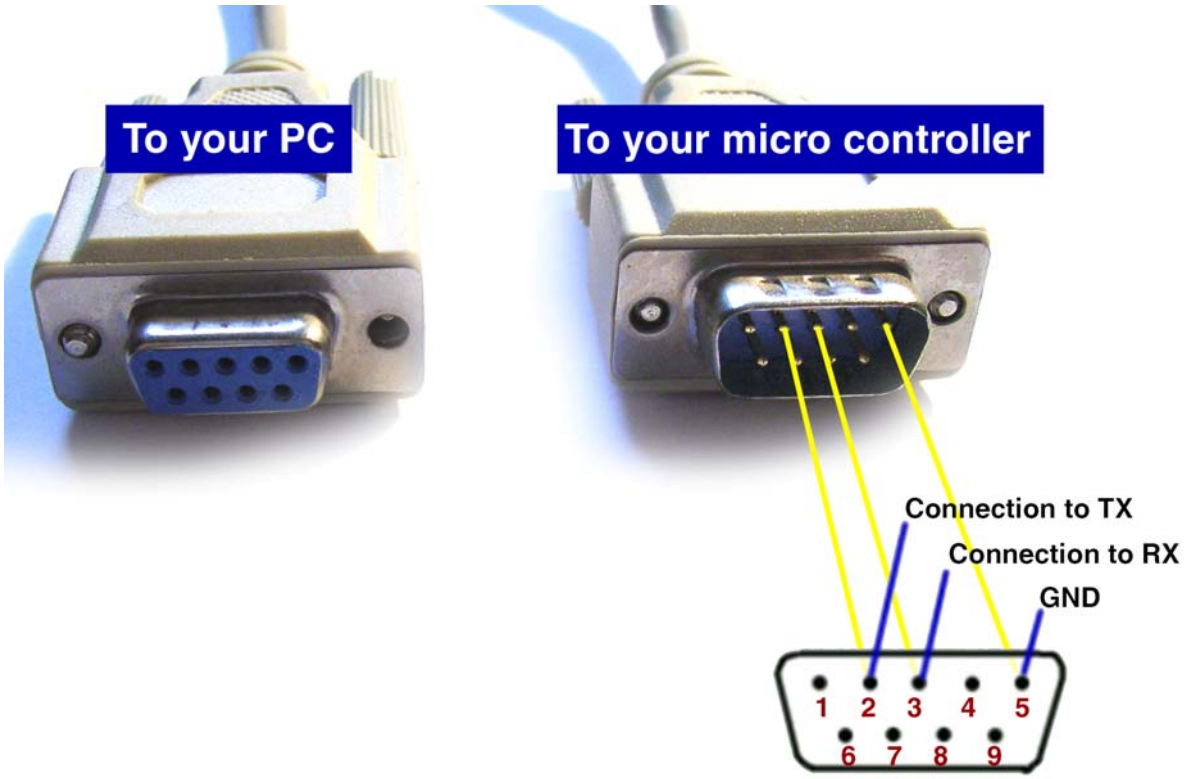
The electrical specification of RS-232 is quite robust, all outputs must be able to sustain a full short-circuit and all inputs must have a schmitt-trigger action. This makes a full-standard RS232 port on a PC much less vulnerable than a TTL-level parallel port.
RS-232 is an a-synchronous protocol, meaning 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 connecter 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 be 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 microcontroller. 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 on our board offers two separate RS232 interfaces, where UART1 can be used without limitation and UART0 might be in conflict with a ISP programming interface as both are using the same controller ports (E0 and E1).
Jumper J11 and J12 will connect the ports E0 and E1 to either a RS232 or a RS485 transceiver. If you want to use ISP programming AND serial interface 0, you need to solder a manual switch above the jumper J11 to open the transceiver connection for reprogramming the board with an ISP programmer interface.

The serial interface 1 is located at the ports D2 and D3 of the ATmega. These two ports (D2 and D3) are connected to the RS232 interface chip at the board through jumper J7 and J8 and this chip is connected to the Rx and Tx Pad. The chip decouples the high voltage RS232 signals from your PC.

If you want to use the RS232 interface, the following example might be helpful to 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 / RS485 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
```

RS-485 Interface

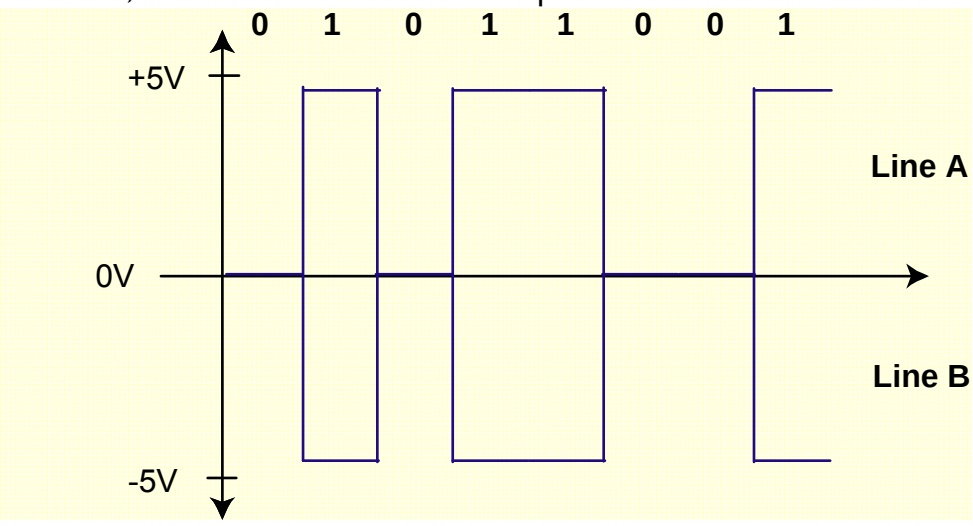
The RS-485-Interface will not need any change in your software. The serial idea is identically, so anything of the above information is valid (but voltages and the wiring). This interface differentiates itself from RS-232 by the used voltages.

As mentioned above, RS232 is a communication standard which sends the data bit for bit at a two-signal-level:
- a voltage of -3 to -25 Volt corresponds with a logical one (1)
- a voltage of +3 to +25 Volt corresponds with a logical zero (0)

The RS-232-connection is set up by a GND-line and a data-line which usually will have either +12 Volt or -12 Volt against GND. The large disadvantage of this standard is the vulnerability to glitches. With a long wiring a glitch is being caught quickly, will be recognized as a bit and will result in a wrong received data.

With RS485 a different voltage level is used. The data lines are called A and B. One bit is represented by level shifting on both lines. A zero (0) is represented by no voltage on any line, a one (1) is represented by a +5 volts level on line A and a -5 Volts level on line B. As it is very unlikely that a glitch will produce a negative level on one wire and a positive on the other wire at the same time, this connection is much more proof than a RS232 connection.

There is no connector-standard as with RS-232. You just work with a 2-wire cable (half duplex) or a 4-wire cable (full duplex).



RS232 / RS485 used Crystals / 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 microcontroller 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 clock at 8 MHz, using of the RS232 interface may result in transmission problems. For that reason, you better use an external crystal.

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 microcontroller is a bit slower, with 18.432, you are overclocking the microcontroller. Usually, this will not result in problems as the ATMega128 can easily run at a 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.

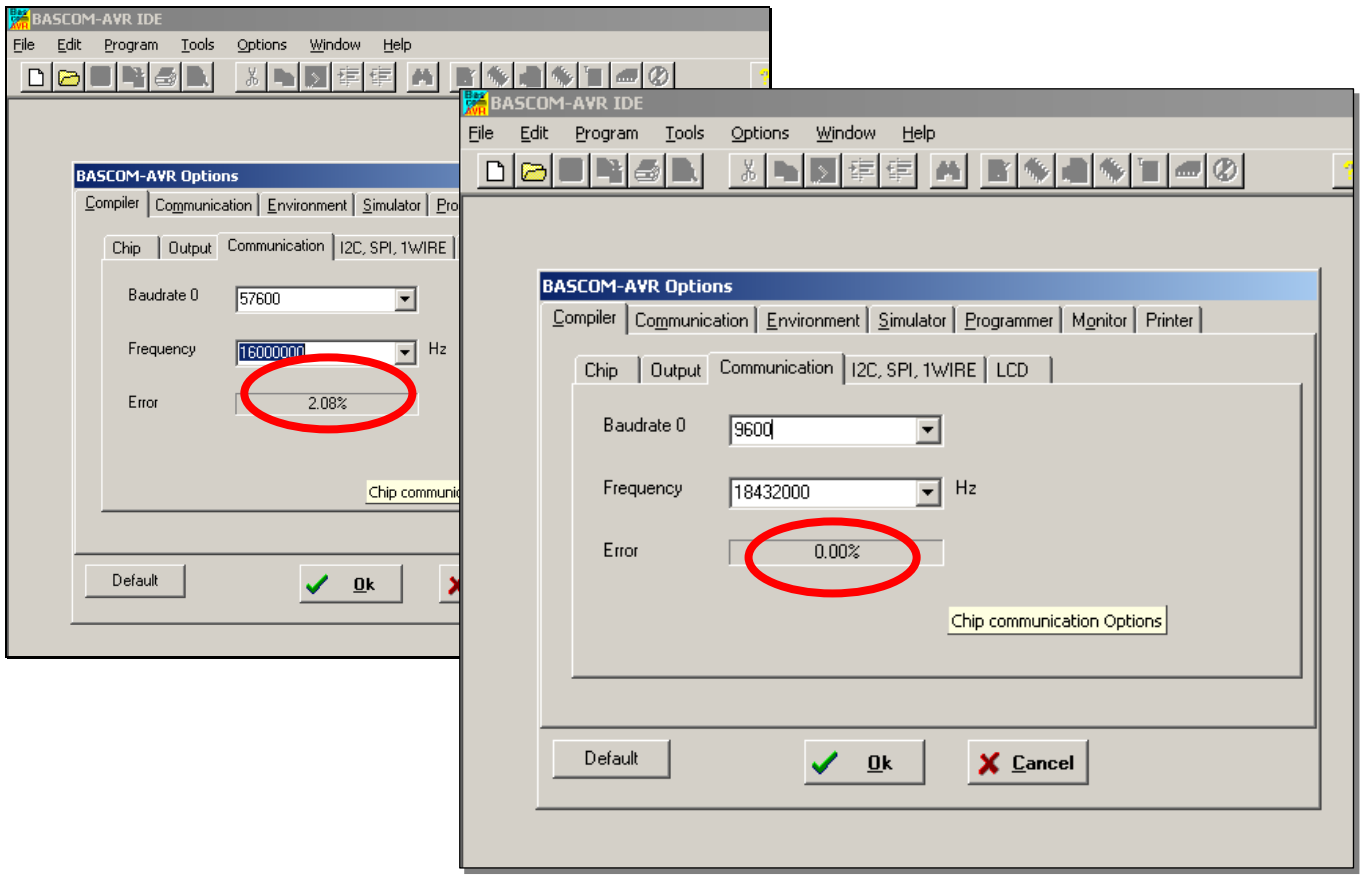
c) You need to tell the compiler, what clock frequency you are providing – otherwise the wrong calculation is done and the transmission will not work. In Bascom® you do this with the command **`$crystal = 8000000`** at the beginning (8000000 for 8 MHz; 16000000 for 16 MHz, 14745600 for 14.7456 MHz etc...).

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

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

	Clock rate of the microcontroller in MHz										
Baud	**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 frequence. 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 very accurate (just a RC-combinaison) 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 like that we will preprogram 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:
Another advantage with a crystal (e.g. 16 Mhz): 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 to 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 microcontroller.

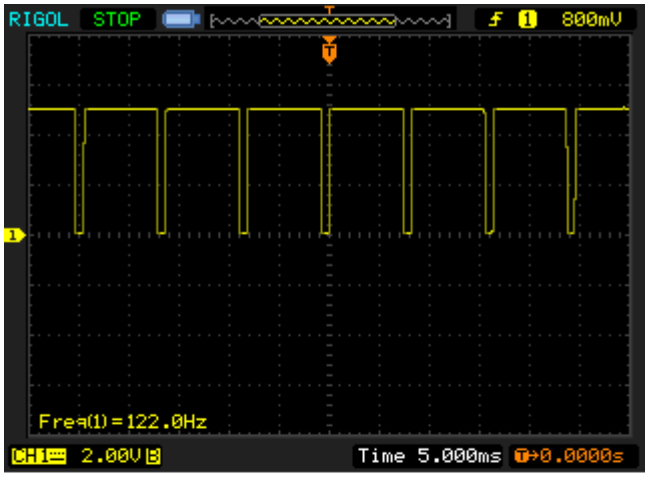
Running (dimming) the display lighting by PWM (pulse wide modulation)

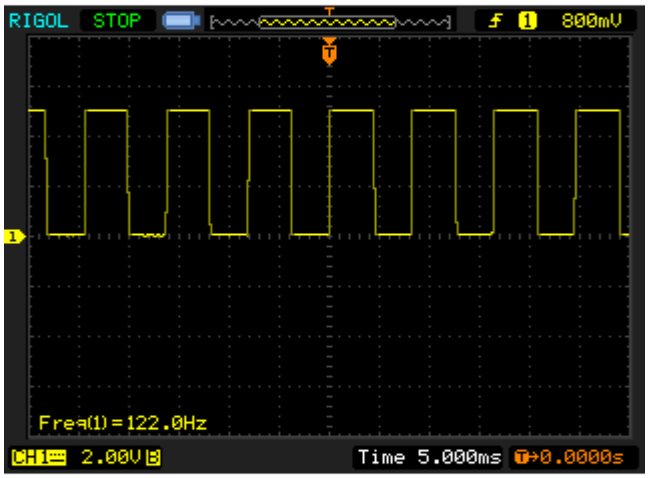
PWM is often being used for controlling the energy usage of a technical system.

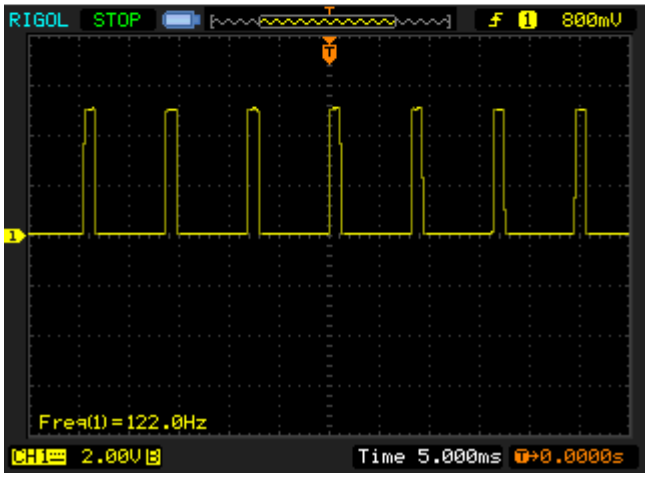
First - some (simplified) theory:
If you switch on and off a LED five times a second for 0.1 second each, you will of course recognize some bothering flickering. But also, the LED was switched off half of the time and will have emitted (and used) only 50% of the energy it might have used if it were switched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

Assumed, you speed up the on/off switching time to 0.001 second 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 being 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 microcontroller is also able to do this and then 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

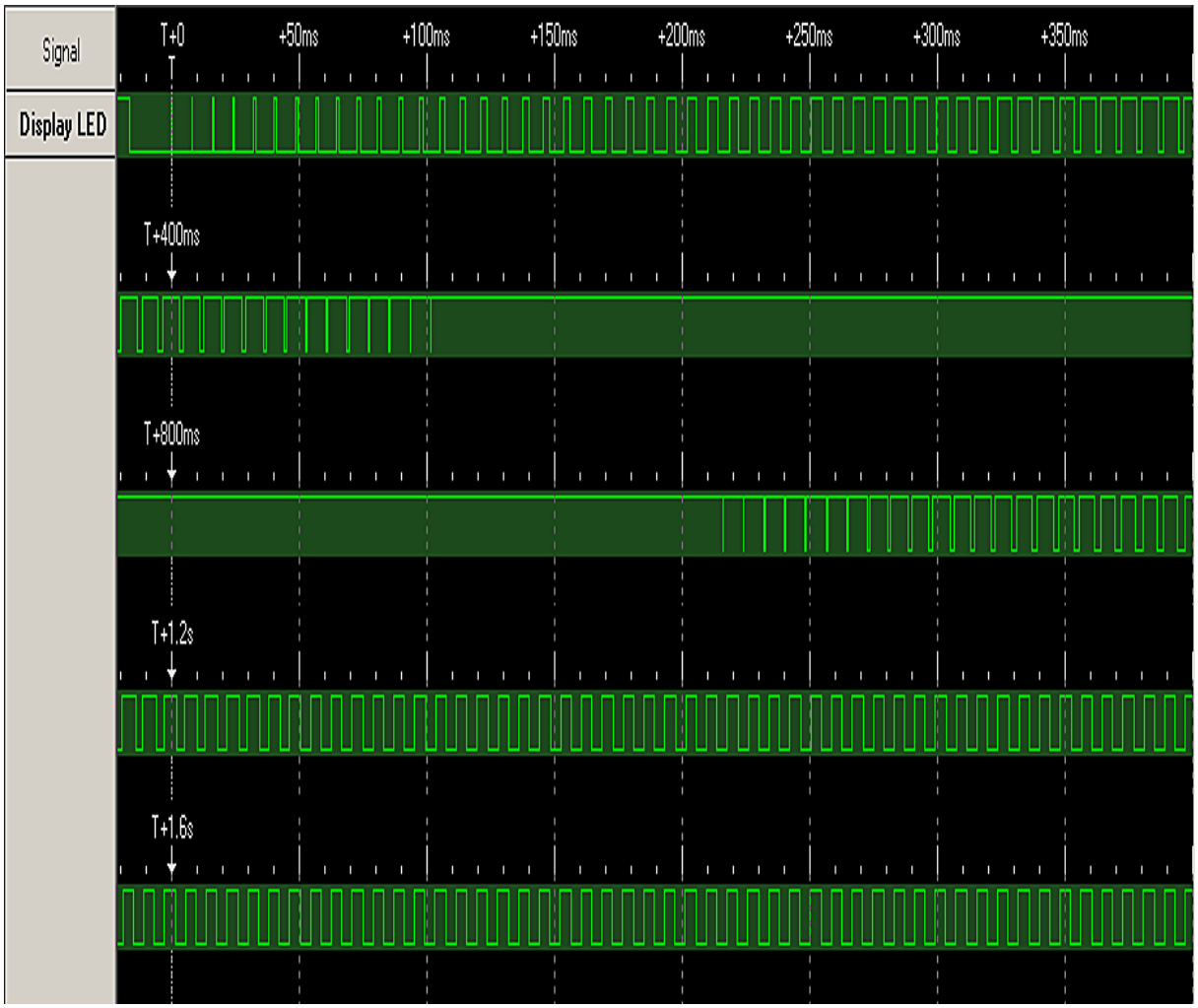
you do not need to give any extra resources to the controller. It will just do the work without using up any of your program space or time. You just need to enable it – then it will run by itself.

An example in Bascom shows this:
The follwing 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.



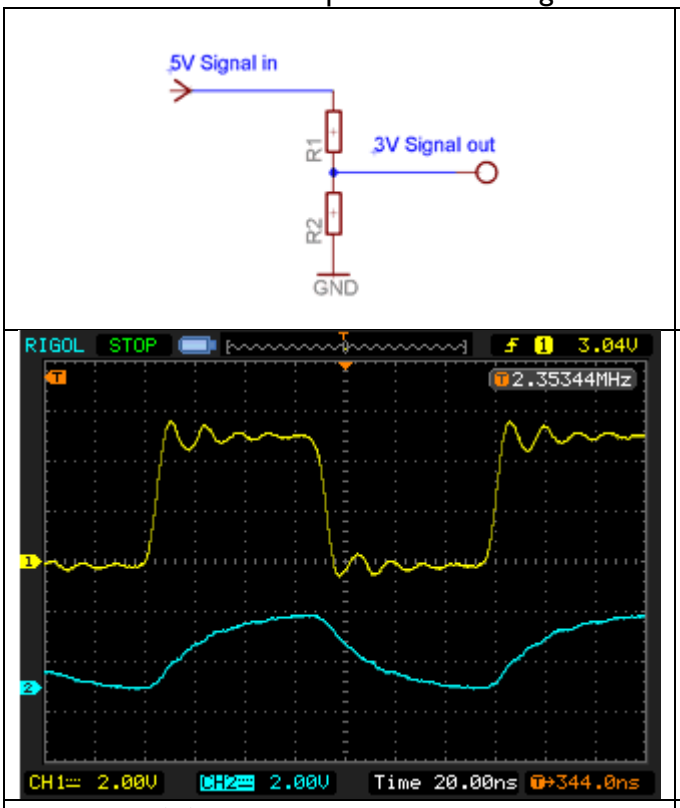
The first 510ms are needed to bring the brightness up from 0 to 100% - 51 steps with 10 ms waiting time each. Then a pause of 500ms (Waitms 500) is visible, during this pause the display lighting is still at 100%. Then, the lighting is being driven from 100% to 50% in 25 steps, and then (at position 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 it does not need any resources.

Appendix: Why the SD card a more complex schematic?

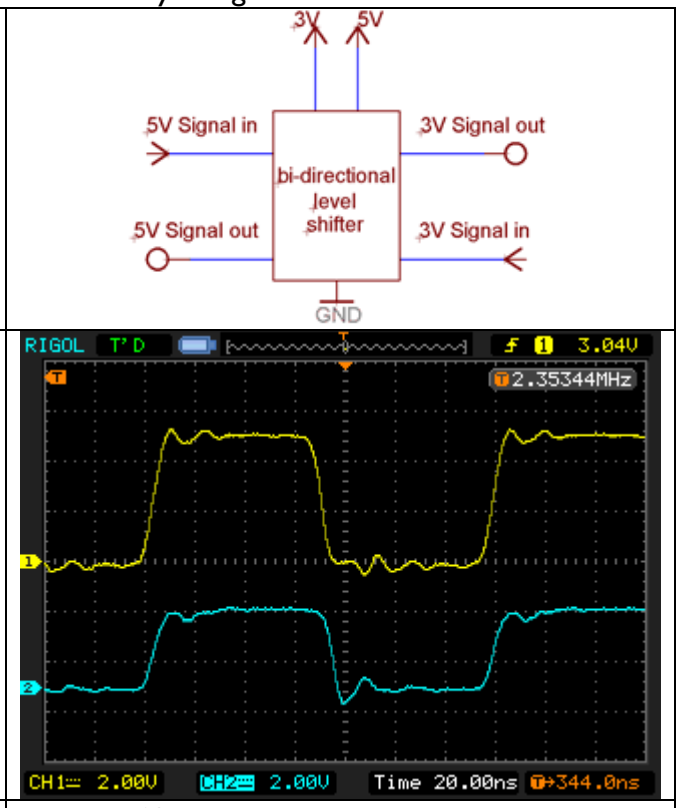
A SD memory card needs a voltage between 2,7 to 3,6 Volts – the needed current fluctuates, depending on the card and usage from less than 1 mA up to 100mA (for a short time). For this reason, solutions with resistors (voltage deviders) or diodes are not usable and even could be dangerous for the SD card. The only professional and long term solution can be a dedicated 3V voltage regulator as it is prepared on our module.

Level shifting
A direct connection between a 5 Volts controller and a SD memory card cannot be achieved as the SD card would be destroyed immediately. The signal level needs to be transformed from 5V to 3V first. A microcontroller running with 3 Volts can be connected directly of course – but even then, our module has its advantage due to the tristate outputs.

Why levelshifting (5V -> 3V) by a voltage divider (two resistors) is difficult
Some people think a level shifting from 5V to 3V can be done with a voltage divider built up with two resistors. Well, this is Ok for slower signals but as soon as the signal level is changing more quickly (and also the raise time of the signal), this becomes critical. The left picture shows the original 5V signal and the input signal at the SD card behind a voltage divider. This is no square signal anymore….. no wonder, this solution creates problems everywhere. For the photos, we are using a voltage divider with 5,6K and 3,9K - with lower values the signals will look a bit better but the current needs is larger of course. Also a slower frequency would help…. but anyway: there is no way to create clean 3V signals behind the voltage divider. Then look at the picture on the right which was made by using a level shifter.



Voltage divider:
Square input – almost sawtooth out



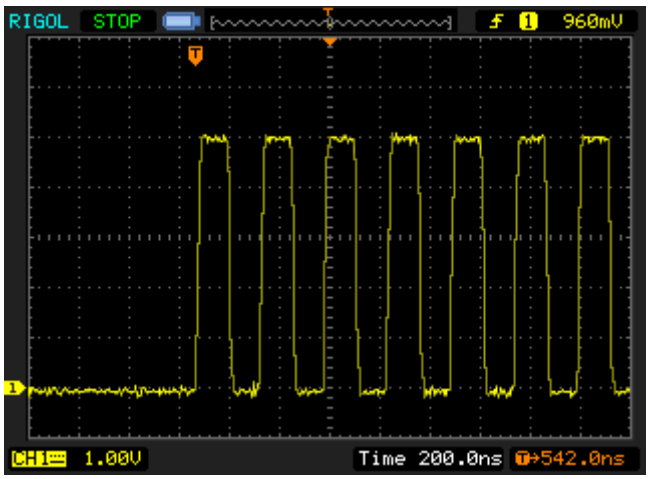
Level shifter:
Square input– square output as it should be.

Levelshifting by transistors

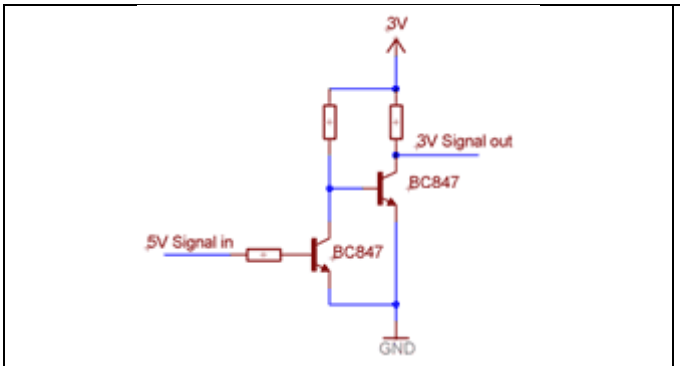
You might now say that a levelshifting from 5V to 3V can be achieved by transistors. This is correct, but you will have big problems to create a working levelshifter for quick signals such as 4 or 8 MHz. Our opinion: there is no affordable solution with transistors which will create clean 3V signals from a 8 MHz signal.

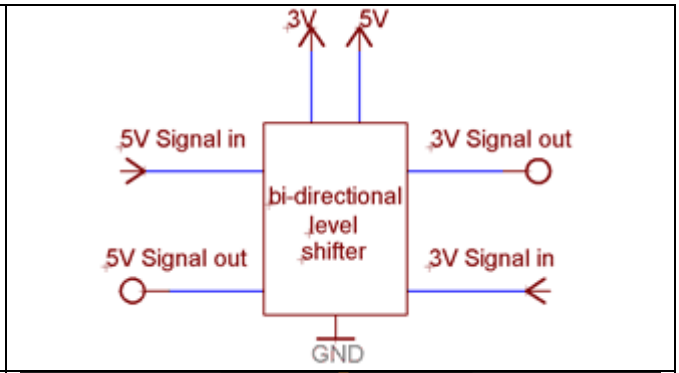
The picture on the right shows the input signal. The pictures at the bottom shows the output signal from a schematic created with transistors (left) and one created with the specialized level shifter.

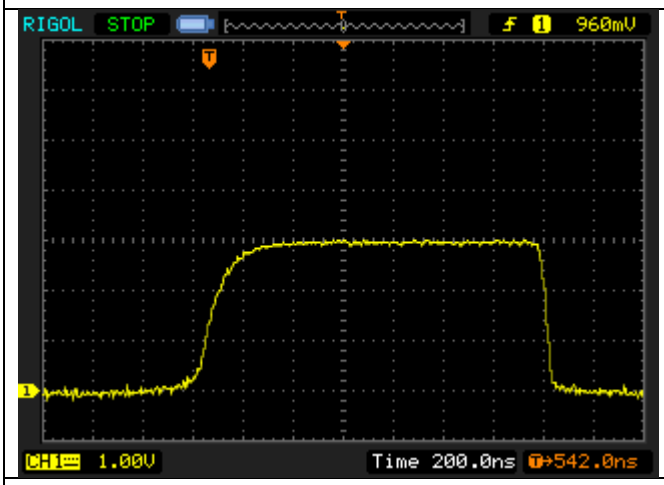
It is clearly visible: Transistors are just not fast enough for quick changing levels and therefore not usable for the levelshifting for a SD memory card (you can recognise: instead 8 clock signals the schematic outputs only one signal as the transistors are not reacting quick enough).



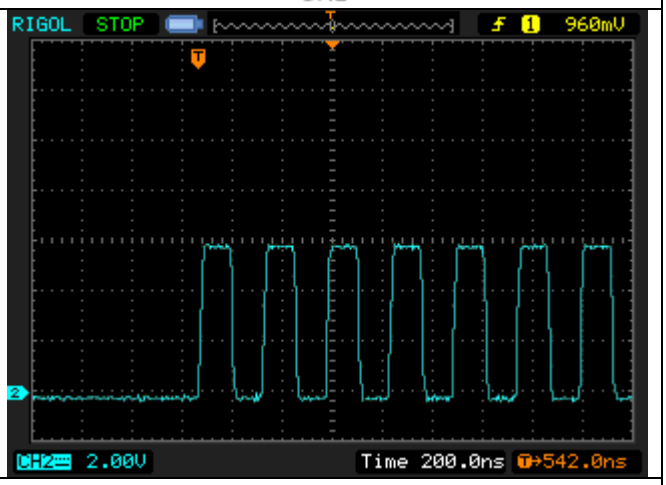
Input signal 5V







Output signal of the transistors – no bits anymore –



Same signal at the output pin of a specialized level shifter

Level shifting in the opposite direction (from 3V to 5V)
The SD memory card not only receives data, it also sends data towards the microcontroller. As the SD card runs with 3V, the output level only is 3V. Theoretically, you may contact the data out signal of the SD card with the data in line of the microcontroller. The 3 Volts level is high enough to be recognized as a high level, even on a 5 Volts microcontroller.
But this is very dangerous – you need to be extremely careful: If you accidently have a 5 Volts signal on this line, your SD card will become destroyed. This can happen either because the programming interface is also connected to the SPI bus (as on most Atmel AVR microcontroller) or you have another SPI bus device working with 5 Volts which sends data, or because you accidently configured the data line of the microcontroller as an output signal – then you have 5 Volts at the data output of the SD card which then immediately turns into a piece of junk.
Thus, a reliable and long term solution can only be a dedicated bidirectional level shifter which protects and shifts levels from 5V to 3V and vice versa.

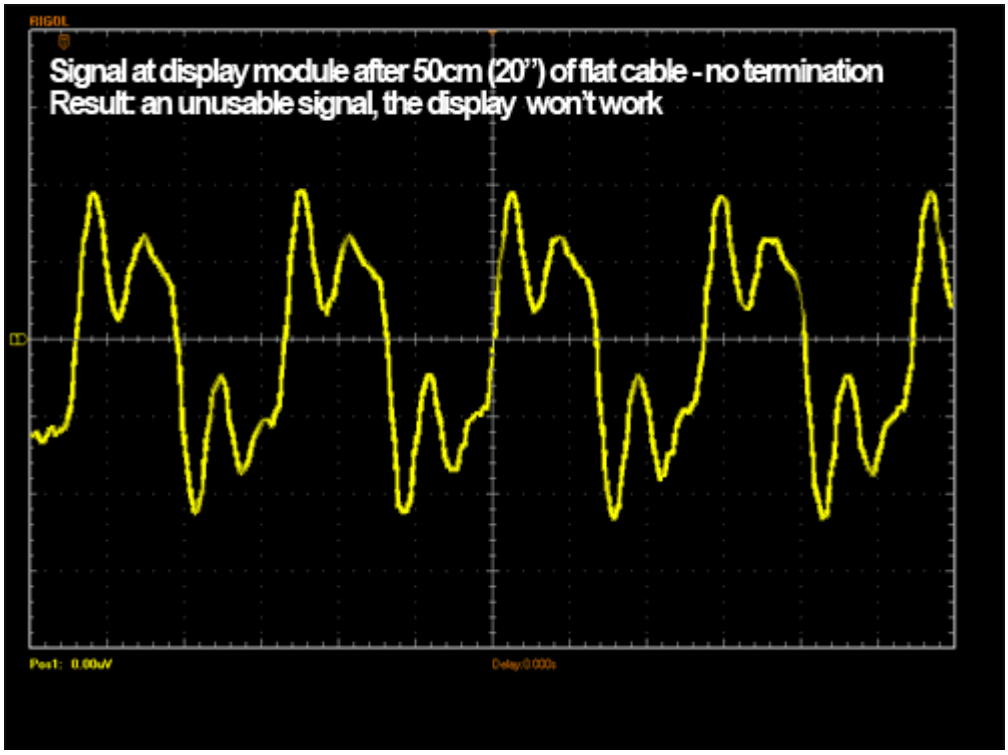
Tristate-Outputs are important!
The world not only consists of your microcontroller and the SD card. Often other devices are using the SPI bus such a display, a touch screen controller or others. If one of these devices is bitchy, it keeps a line fixed to a high or low level after a transmission. Then all other devices on the SPI bus are not able to communicate with the microcontroller. The SD card is one of these tricky devices which hold the data-out-line to "high" after the transmission – the display for example cannot be used then anymore. For this reason, we have chosen a level shifter with tristate outputs. By setting the line "Active" to high or low you are able to connect or disconnect the SD card completely from the SPI bus. Thus, the card cannot be disturbed by other devices and also cannot disturb others by itself.

Appendix: the possible need of termination

External parts, driven by the microcontroller with high speed data transfer, are often showing problems receiving the data correctly.

Even an 8 bit Atmel can run data over the wires with 8 Mhz which will cause in maximum cable length of 15-20 cm (approx. 6-8 inch), everything longer will result in signal reflections at the device, thus a reliable data transfer will not be possible due to a bad signal quality.

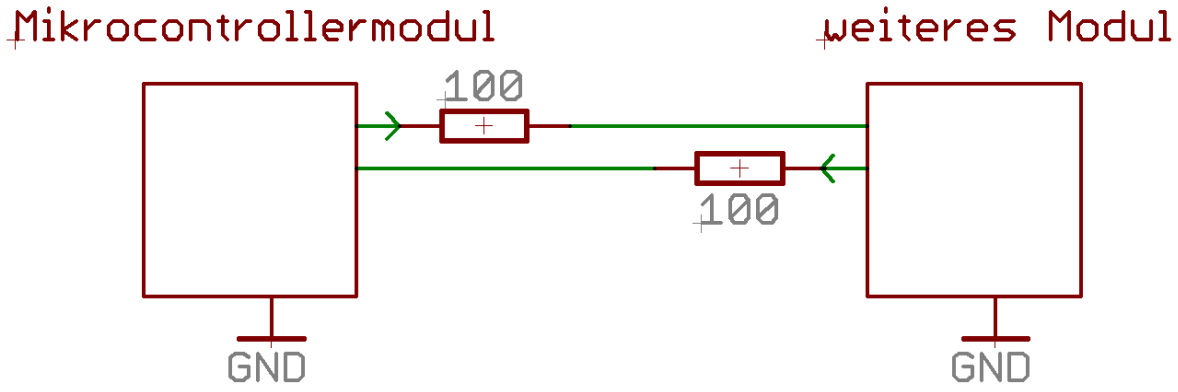
The picture below shows a non terminated signal line on a 50cm (20 inch) cable. It is obvious, that this low quality signal with lots of ringing will result in a non-reliable connection.



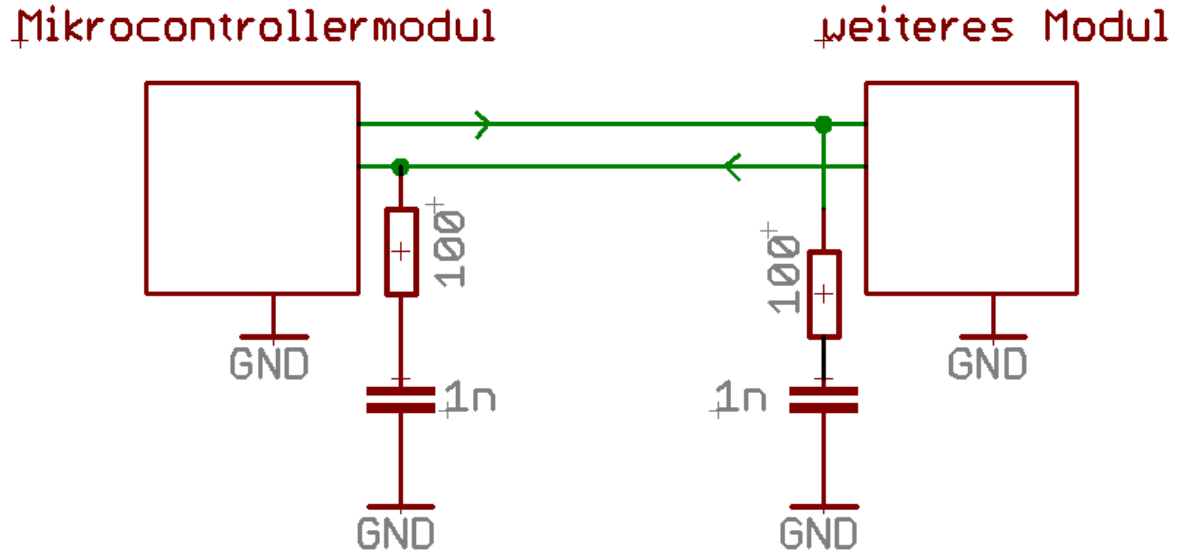
Instead the once clean cycle signal, the display gets some …. well …. sub optimal signal.

There are two possible options for find a remedy

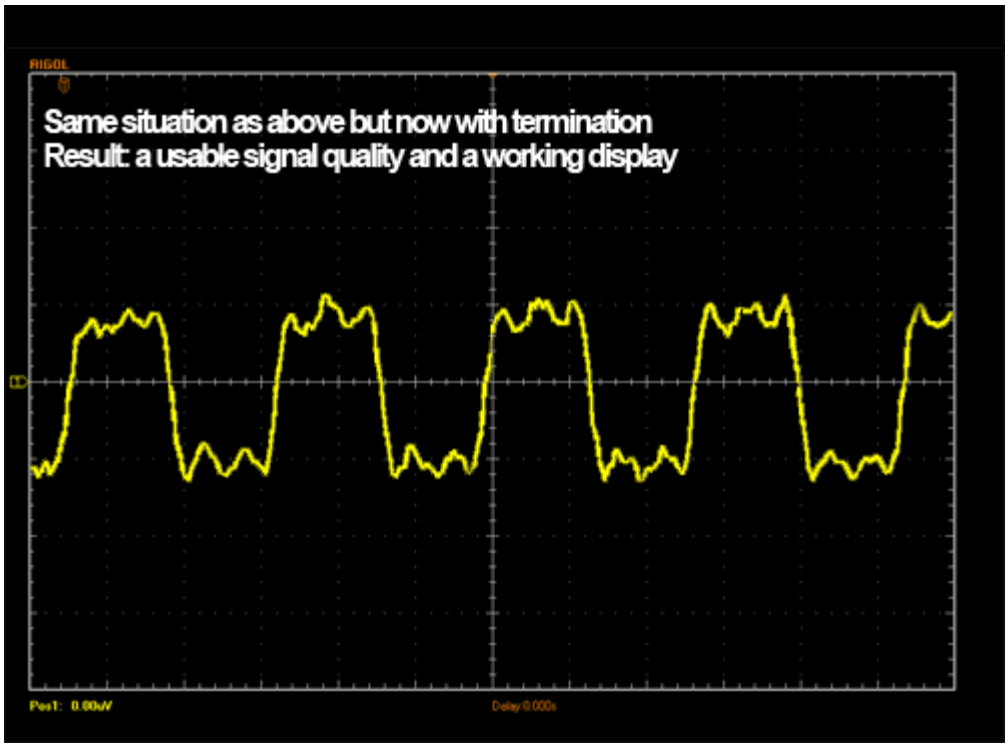
Option 1)
All signals will get a serial resistor added at the sender (thus the microcontroller). This serial resistor shall be around the impedance of the used cable and has to be determined by conducting an experiment (check clock signal at display module with an oscilloscope) as each configuration needs it specific resistor. E.g. a ribbon cable will probably need a resistor between 80 and 120 Ohms.



Option 2)
Each signal will be terminated at the end of the device (thus the display module). This, by the way, can also be done together with option 1. In this case an AC termination with a resistor and a capacitor is the preferred action.



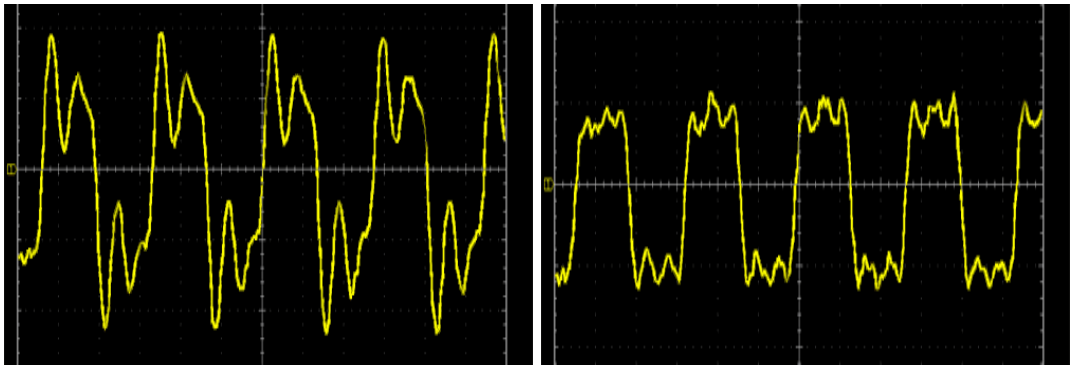
The picture below shows exactly the same configuration as before, but with a terminated line at the display module. The signal is not 100% perfect, but good enough for the display as high and low can be easily detected.



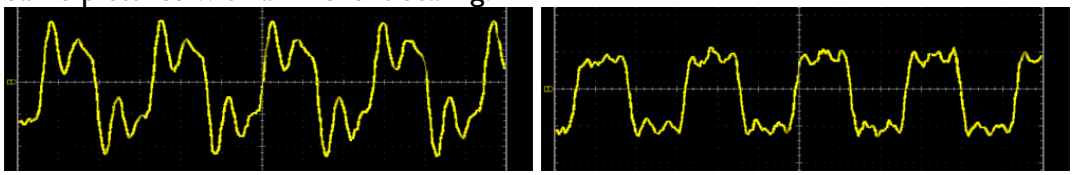
The exact part dimension need to be find out by experiment as each cable type and cable length need different values. However, good values to start with are a 1nF capacitor and a 100 Ohm resistor.

Terminating on the receivers side also has the advantage that you are creating a low pass filter which eliminates any glitches. Also the cable will emit less interferences (important for EMV).

Now, both oscilloscope pictures are shown side by side to makes it easier to recognize the dramatic improvement of the signal quality.



Same pictures with a different scaling:

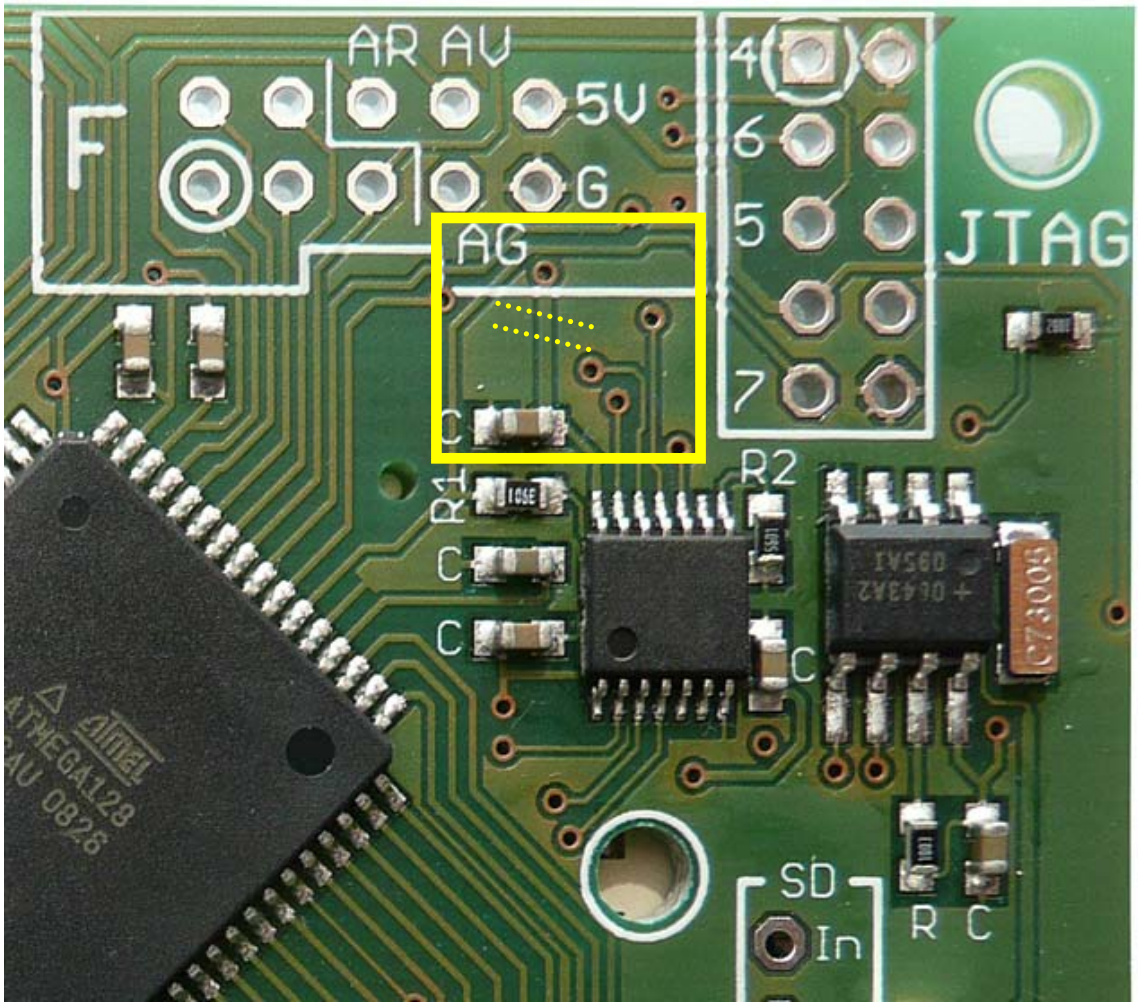


Appendix: Errata

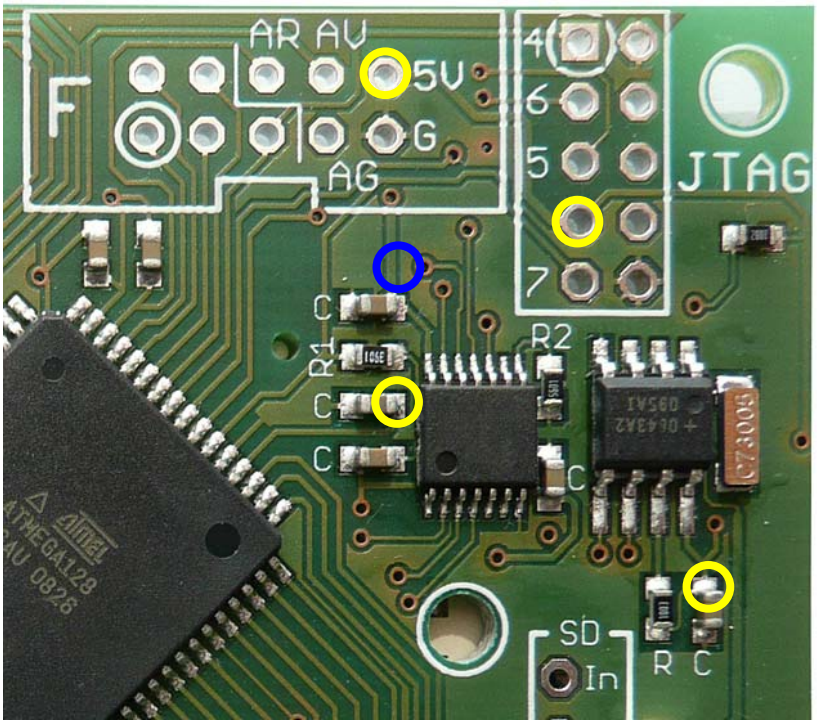
A) Switch-off of the bidirectional levelshifter for SD cards

A switch-off of the levelshifter is not possible. As of the low current need of this part (1µA) this causes no trouble (see also page 52).
If you purchased a board before mid of August 2009 with the two options „switch off“ AND „SD card module“ (or if you enhance your board later by yourself) you need to do the following correction. At boards, ordered with both options, we are doing this correction since mid of August. If you are not using SD card module together with the switch-off option, the changes are not needed:

a) Disconnect the VCC from the levelshifter from the VCC5_2 line (switched 5V line):
Cut with a sharp knife (or scalpell) the connection above the right solder pad of the capacitor shown below (the dottet line shows a possible cut-area).



b) Connect VCC of the levelshifter with the permanent VCC5 line.
The right pad of the capacitor (blue circle) need to be connected with VCC5 now (e.g. with a short wire). VCC5 is available at all yellow marked areas.



Hint: quick and easy this can be done by soldering a low value resistor (approx. 10-20 Ohm) between the two capacitors as shown as the photo below.