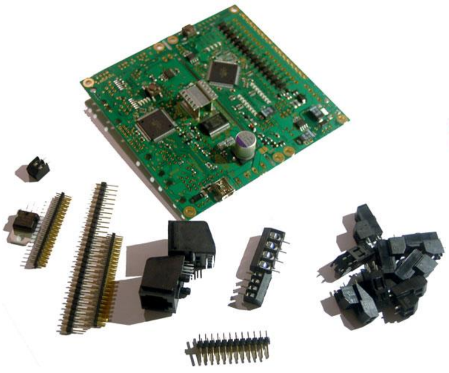


Manual "Setup"

SMD-stocked kit

OpenDCC GBM

Option 1 - as a feedback system

GBMboostV1.6
GBMboostV1.8
GBM16T

Kit suitable for: ANFÄNGER

Table of Contents

Introduction:

This manual describes the operation of SMD assembled kit for OpenDCC GBM from the DIY OpenDCC series.

Read these instructions carefully before beginning assembly and follow the safety instructions.
The assembly and the use of the smallest electronic components requires a significant level of experience, especially with the use of advanced SMD components.
This guide does not pretend to be a commercially manufactured product. It only serves as an aid to construction of the kit for experienced modelers and interested solely for DIY.

This manual has been carefully checked and prepared in good faith.
There can be no claim to completeness, timeliness and accuracy are collected. If trade names or protected names are used, all rights reserved by the owners.

We assume no responsibility for any kind which would be derived from the use of this manual, the contents or their use. The user of this manual is agrees by using this manual.

As used herein, and described in some software can be used on the website of www.opendcc.de for download, extended and improved. For more about the use of software, hardware and application is described on the website of OpenDCC Fichtel and train. The user and the user agrees to the rules described there without reservation.

Commercial use of the Software or any part thereof is not permitted!

These instructions shall not be used for any other use except the intended use for the construction of OpenDCC GBM.
Other use requires the permission of the author or the copyright owner of the website and www.opendcc.de and www.fichtelbahn.de.

Safety:

The instructions described in this module is an electrically operated device.
You must take all necessary precautions in the operation, which with the of electric current are applied.
Create the module not to mains voltage.
Do not use power supplies of PCs. These devices are not grounded, i.e. it can operationally on the tracks and connected devices, high voltages - Danger!
Ground not conductive parts of their model train layout!
All shields, cable shields, etc. are necessary when considered necessary to lead to a common floating point together.
The completed module is to be operated exclusively with low voltage protection and separation.
Model railways are classified into mainstream legal opinion as a toy. Here are special rules.
For current feed exclusively the commercial AC power supplies are to be used with the appropriate authorization.
If you are buying on the appropriate classification of the power supply.
Read more at www.vde.de.

Intended Use:

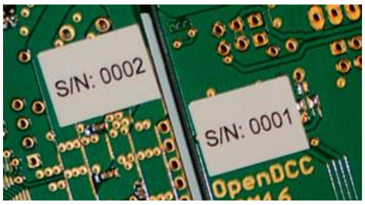
The module is provided for only in model railroad layouts that are to be controlled digitally, to be used for driving, switching and signaling.
Any other use is improper.
The module is not intended for assembled by children under 14 years, to be installed or operated.

Preparation:

- Nothing goes without systematic work.
- equipment and tools ready, clean tips?
- Check the required parts to be complete.
- Follow the instructions step by step and check your work through the control points. Go without successful measurement to the next section.

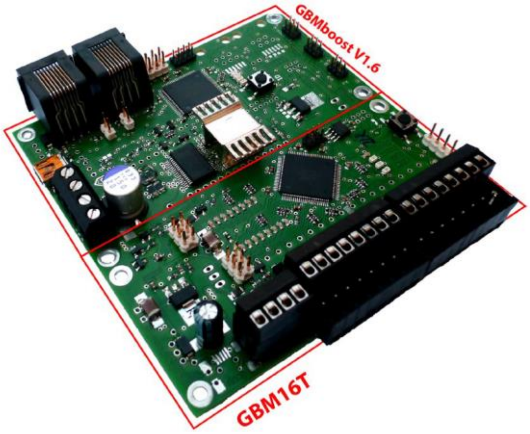
Important note for assembly instructions
This guide is designed specifically for the SMD preassembled kit, then all assembly operations are not necessary.
If you want a more accurate description of placement or to realize the normal OpenDCC kit, you need the normal "Installation and Commissioning" guide. These can be found on the website Fichtelbahn.de.

About the serial number:
The serial number pasted on the back of the module is a hardware serial number and has nothing to do with the unique ID serial number for BiDiB modules. Here are two different serial numbers.



1 Chapter: Information OpenDCC GBM

1.1 Functional Description:



What we put into operation?
The OpenDCC GBM consists essentially of 2 parts:

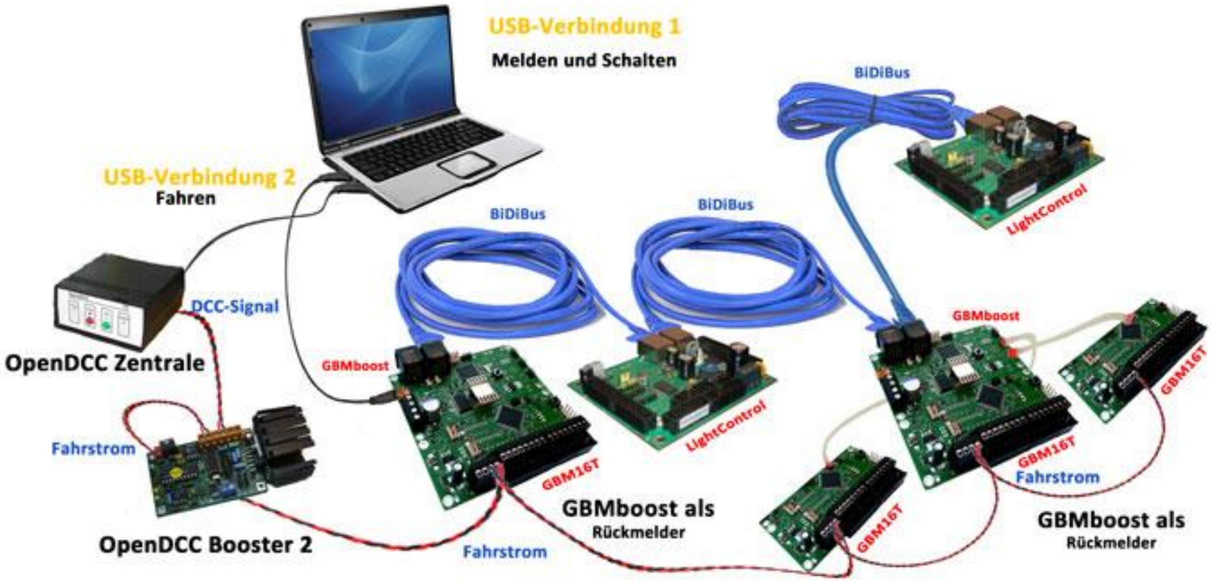
- The **GBMboost** (old name: Controlproc) with connectivity to other modules and the USB interface to communicate with the PC as the master. A DCC power supply module (booster) with on the same board.
- The **GBM16T** (old name: Trackproc) with 16 occupancy detectors.

The close relationship to known GBM16TC is not accidental. The existing components and GBM16TC GBM16T can be reused.

When the concept of GBM OpenDCC GBM16TC changed. The communication between the modules is done exclusively through BiDiB.
Via USB 2.0 connection to the PC is ensured.
More information about Software Master / Node - configuration, functions and applications with control programs, please visit the website of www.opendcc.de, www.fichtelbahn.de and OpenDCC forum. In the application, exclusively, the right choice of software on the desired function.

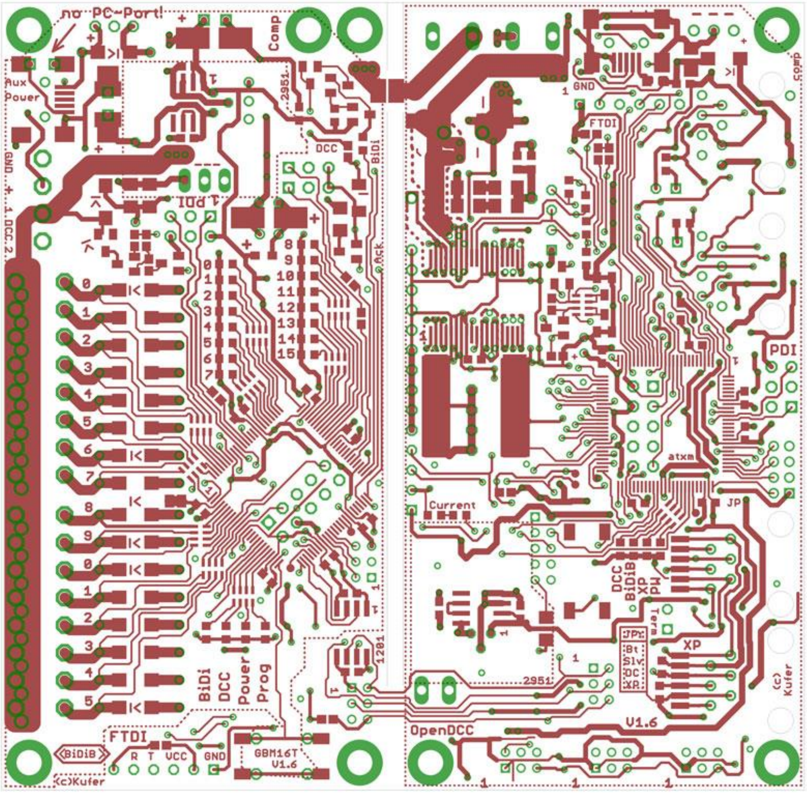
Applications, connecting examples and information about individual software applications are described in detail in a separate document.

And might look like an application with BIDIB "Version 1":

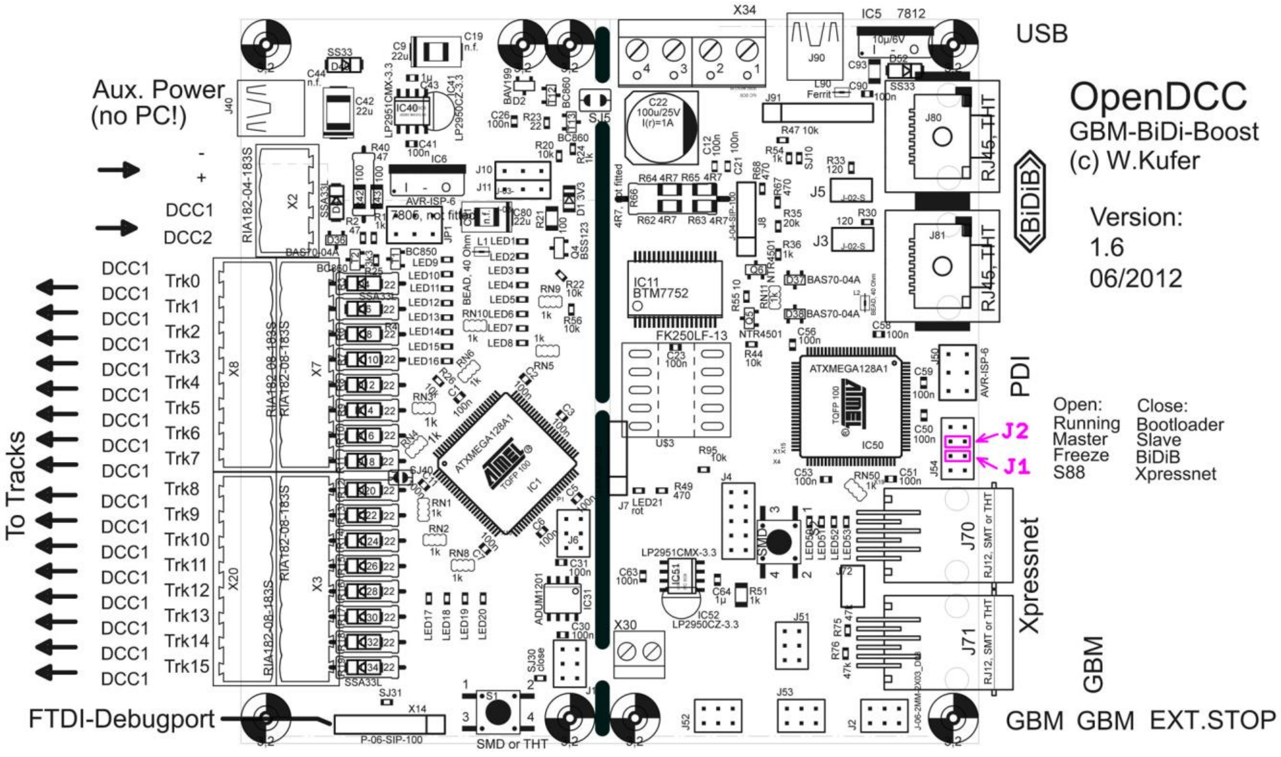


1.2 Schematic, layout and assembly:

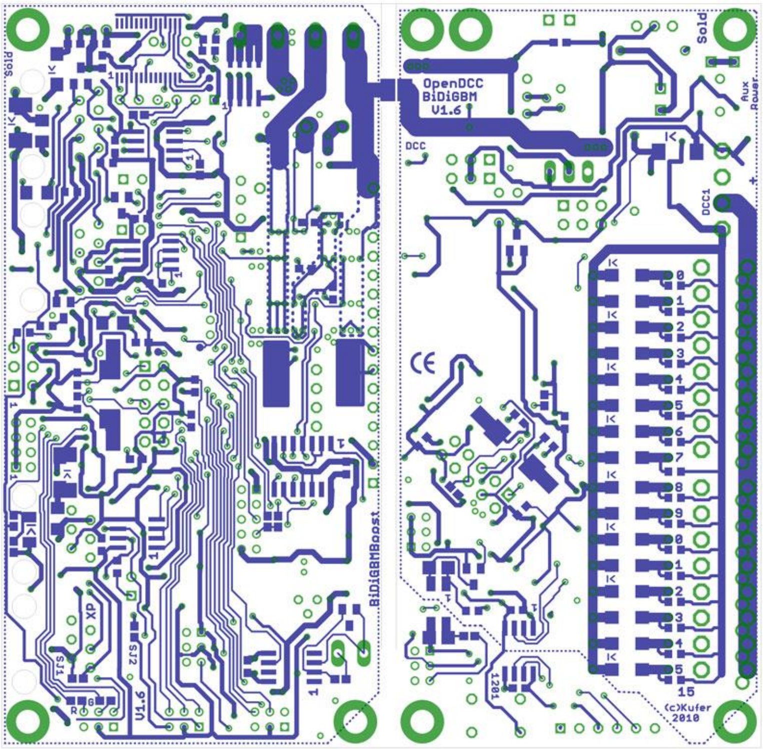
Layout component side



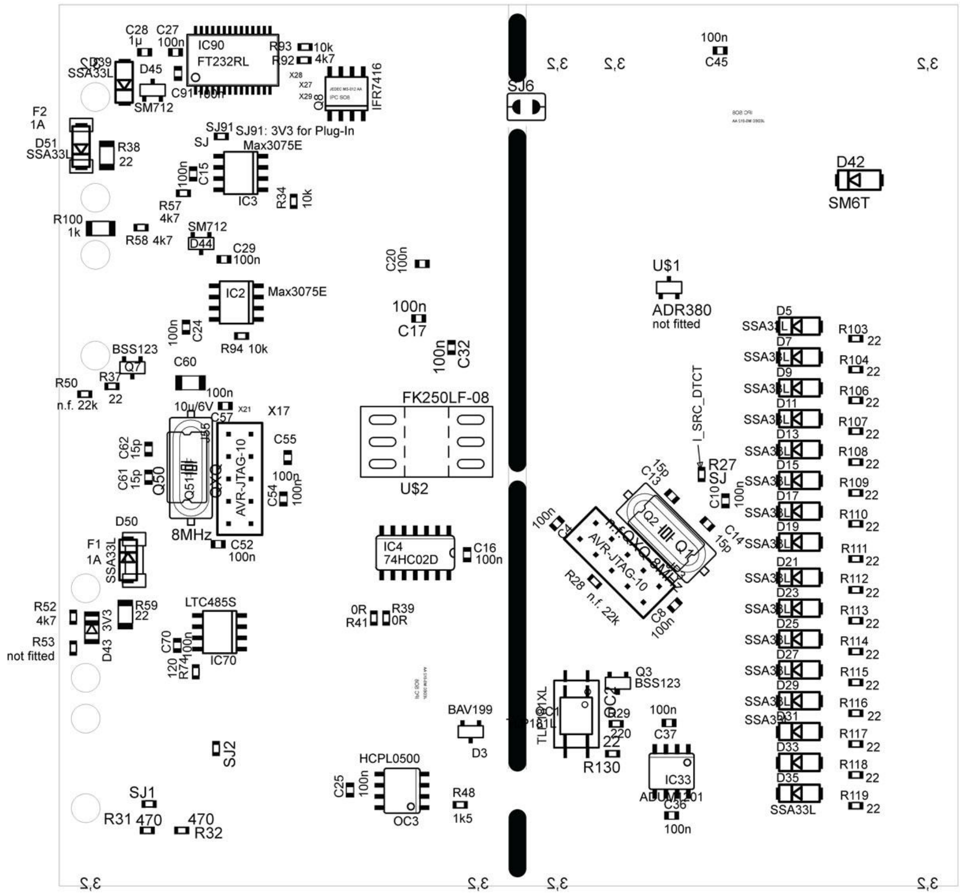
Placement component side



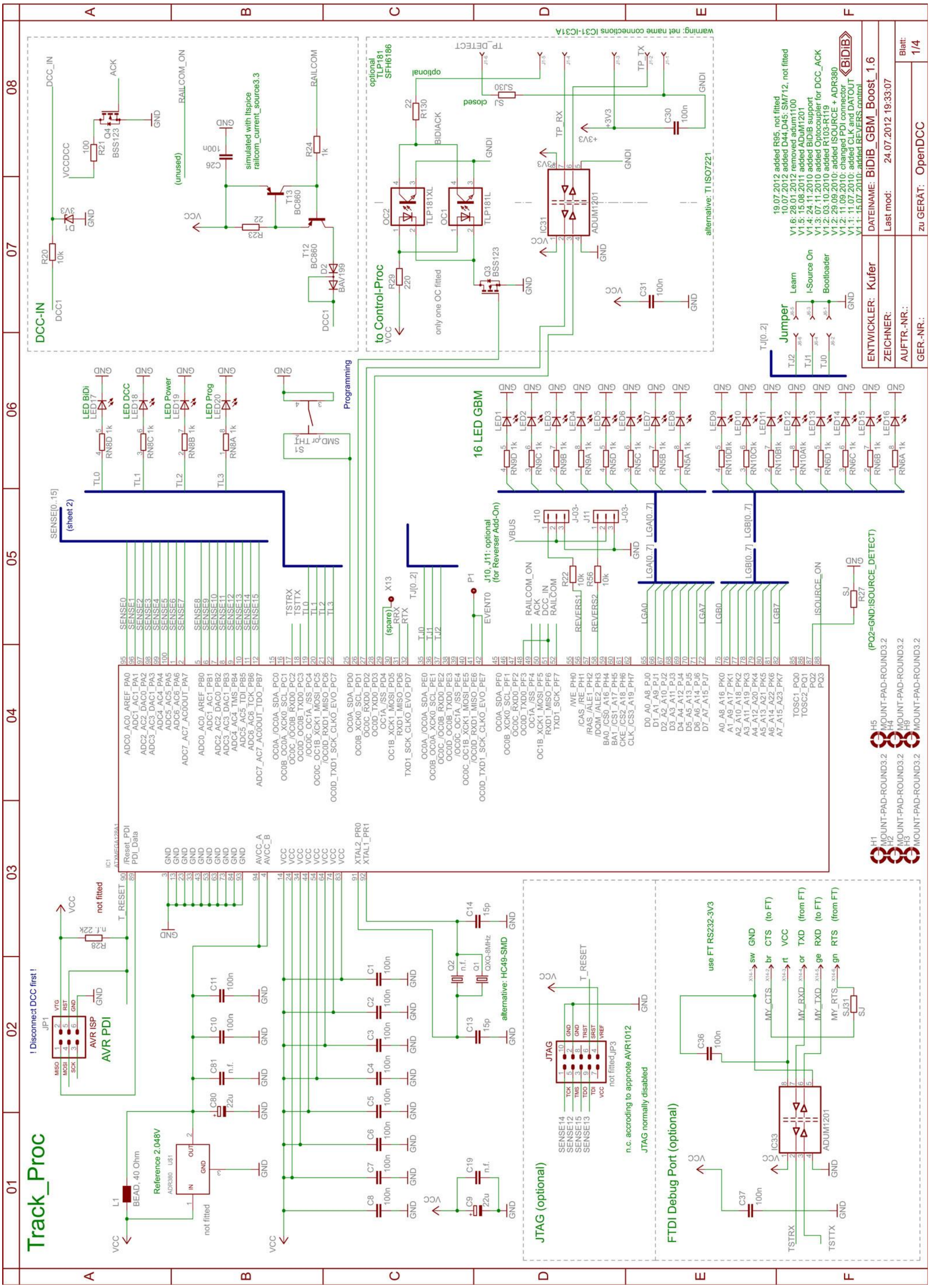
Layout solder



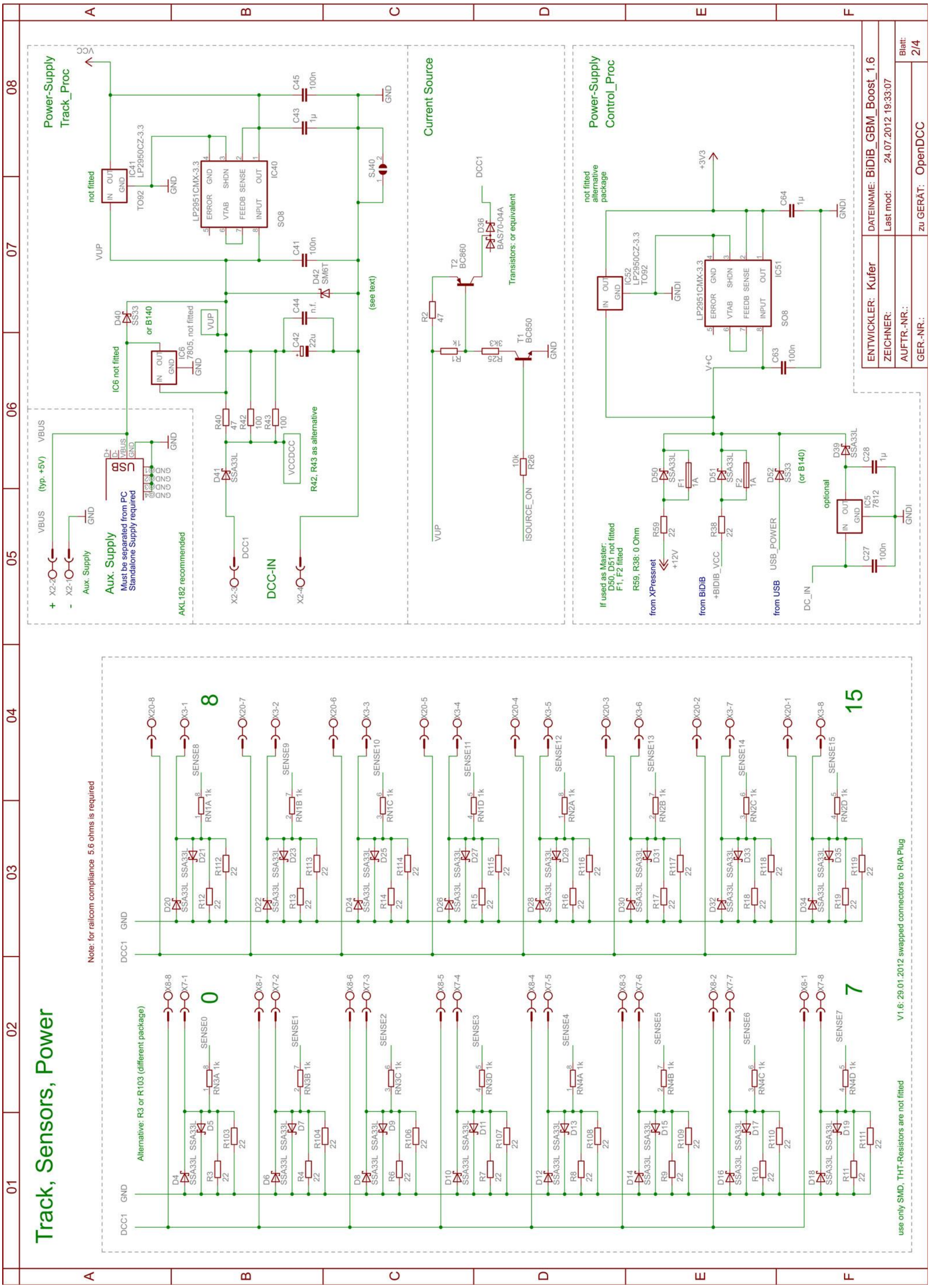
Placement solder



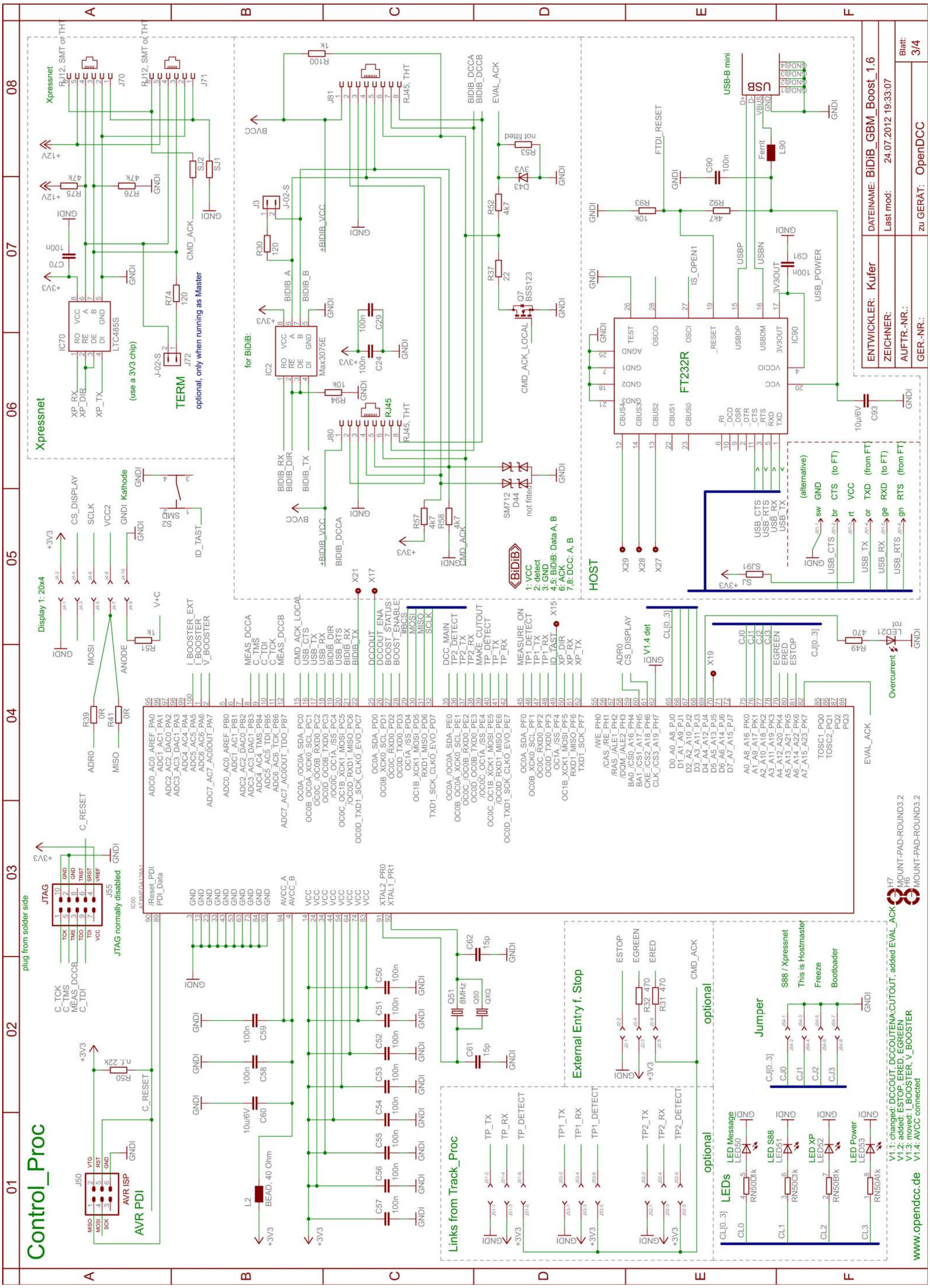
Schematic Page 1


Track_Proc
DCC-IN
to Control-Proc
16 LED GBM
Programming
Jumper
JTAG (optional)
FTDI Debug Port (optional)
AVR ISP
AVR PDI
! Disconnect DCC first !
Reference 2.048V
alternative: TI ISO7221
alternative: HC49-SMD
warning: net name connections IC31-IC31A
simulated with ltspice railcom_current_source3.3
J10, J11: optional (for Reversier Add-On)
(PQ2=GND:ISOURCE_DETECT)
use FT RS232-3V3
n.c. according to appnote AVR1012
JTAG normally disabled
only one OC fitted
LED BiDi
LED DCC
LED Power
LED Prog
ATXMEGA128A1
ADUM1201
19.07.2012 added R95, not fitted
10.07.2012 added D44,D45: SM712, not fitted
V1.6: 28.01.2012 removed adum1100
V1.5: 15.08.2011 added ADuM1201
V1.4: 24.11.2010 added BiDiB support
V1.3: 07.11.2010 added Optocoupler for DCC_ACK
V1.2: 03.10.2010 added R103-R119
V1.2: 29.09.2010: added ISOURCE + ADR380
V1.2: 11.09.2010: changed PDI connector
V1.1: 11.07.2010: added CLK and DATOUT
V1.1: 15.07.2010: added REVERS control
BiDiB
ENTWICKLER: Kufer
ZEICHNER:
AUFTR.-NR.:
GER.-NR.:
DATEINAME: BiDiB_GBM_Boost_1.6
Last mod: 24.07.2012 19:33:07
zu GERÄT: OpenDCC
Blatt: 1/4

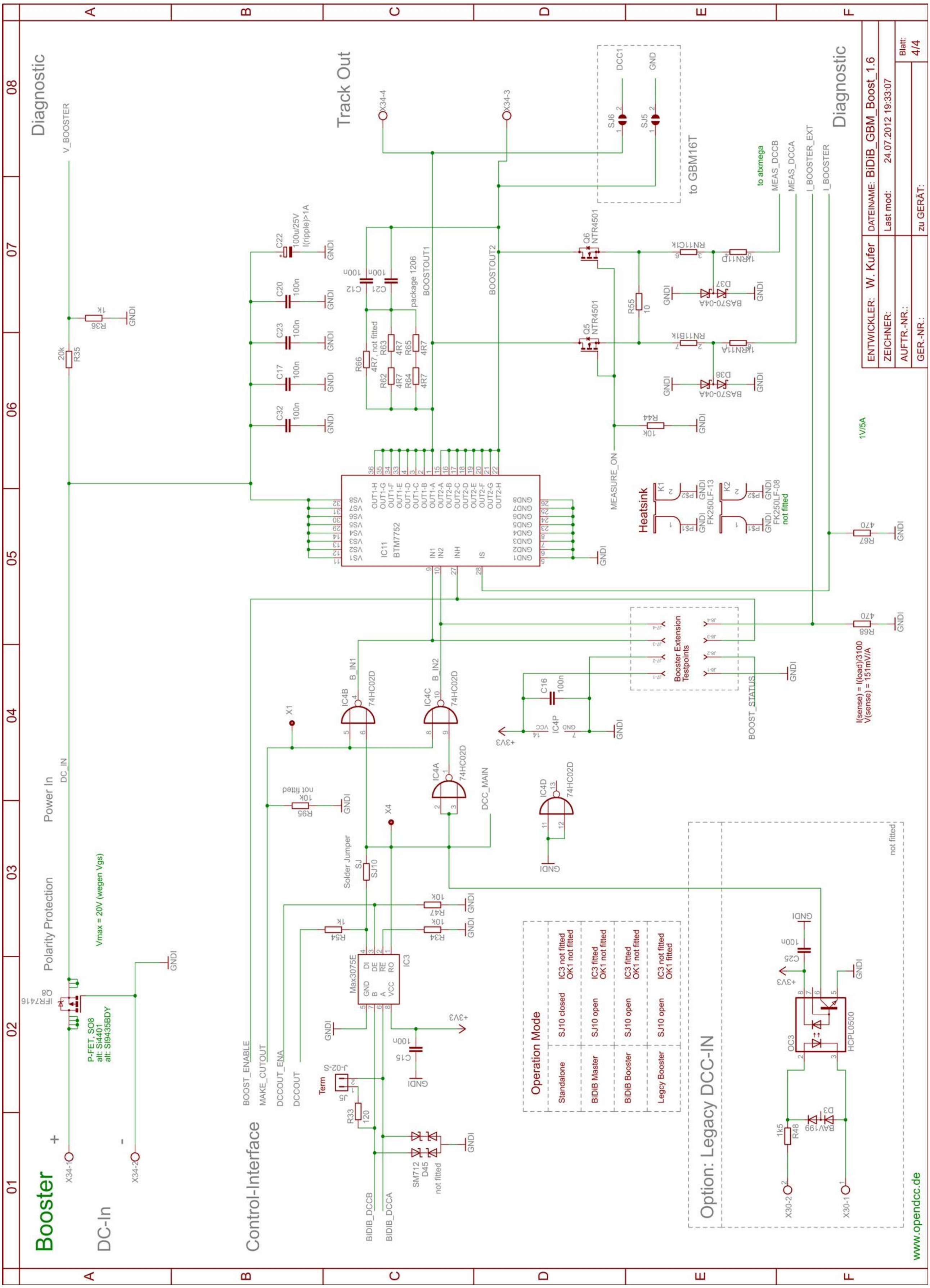
Schematic Page 2



Schematic page 3



Schematic page 4



2 Chapter: Equipping the GBMboost V1.6

Chapter 2 deals with the placement of the remaining necessary components to wire GBMboost.
Furthermore, there are possibilities for the two operating GBMboost (Master / Node), and various combinations of power supply (USB, BIDIB). These settings are discussed in Chapter 2 also.

THT Solder Components:

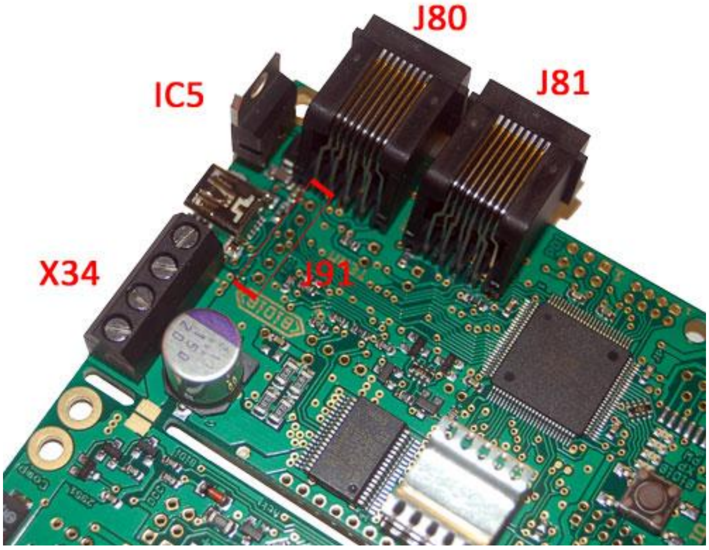
In the bag you will find enclosed the remaining parts to be assembled.

IC5 ... 12V voltage regulator (note direction of installation ... see photo)

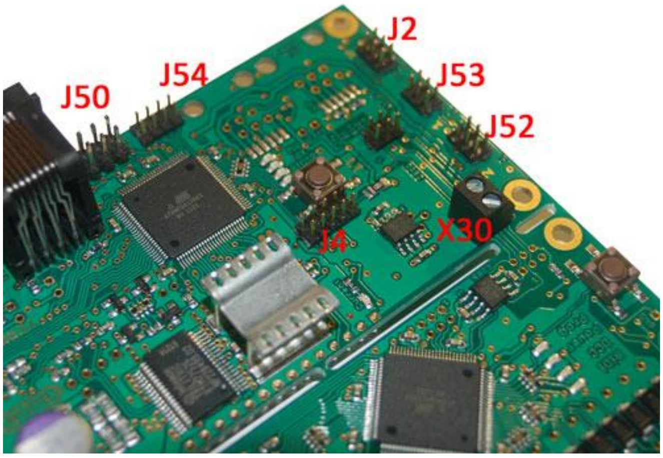
J80, J81 ... RJ45 jack for BiDiBus

X34 ... Terminal for the power supply and DCC booster output

J91 ... Debugging interface of GBMboost (optional)



⚠ The jumper SJ5 and SJ6 need in this variant 1 (occupancy detector) remain open and may **not** be closed. **That would otherwise lead to a short circuit!**



J50 ... PDI programming

J54 ... Settings of jumpers

J2 ... E-stop connection

J52, J53 ... Connection for additional modules GBM16T

J4 ... Display Header

Control Choices (Master / Node):

With the purchase of a master node or device in the Fichtelbahn web shop this point was "operating options" for you already executed and the processor with the matching boot loader and firmware features. This is vital for you as a guide only!

The communication on the BiDiBus takes place on the basis of master and slave.
It has a master at BiDiBus BiDiB and numerous other modules as slave.
These are called Nodes!

So you have to make a GBMboost to master!
Only the master has connecting via USB to the PC!
It may also depend on only one master is BiDiBus!

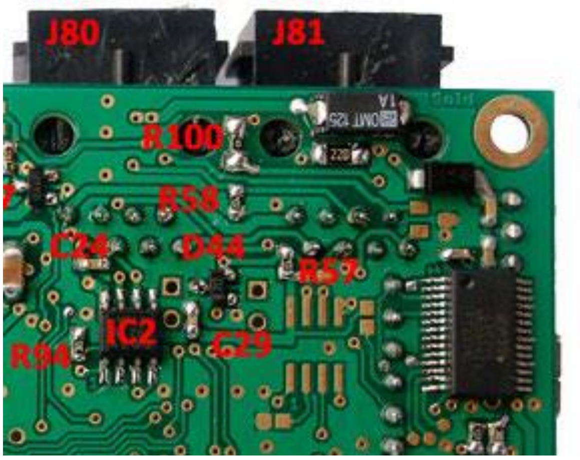
The difference between master and slave

(except for the different firmware):

There are equip on the back of the Master:

R57, R58 and R100 with 4.7 kOhm with 1kOhm

Remark:
The bias resistors R57, R58 can also be equipped with a 1.5 kohm.



The resistors R57, R58 and R100 may only be equipped with master device. Depend at all other than the OpenDCC GBMs Node (Slave Device) on the pads BiDiBus remain unoccupied. The R100 resistor is a pullup resistor for the ACK network. The other two resistors are bias resistors and ensure a stable operating point for the RS485 module.

Supply:

The GBMboost can be supplied from various sources or themselves serve as a source for the BiDiBus. The manual "Installation and Commissioning" kit OpenDCC GBM can be found in Section 2.1, a detailed explanation placement.

In this pre-populated SMD version, the basic underlying combinations already preloaded and you do not need to change anything! This is vital for you as a guide only.

Your SMD-tipped GBMboost is prepared for:
Powered by USB, the BiDiBus and an external power source

When using one of these sources no conversion is necessary!
But you should keep in mind the power capacity!

Powered from USB: (max. 500mA)
Only suitable for a module interface or track occupancy detectors without booster function.

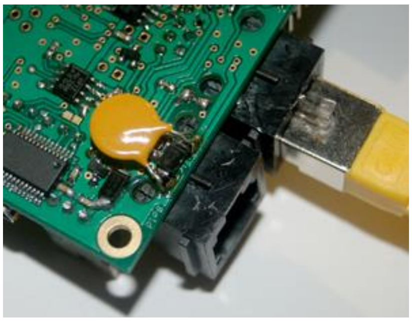
Powered by the BiDiBus:
Only suitable for track occupancy detectors without the booster function. The requirement that an GBMboost feeds the BiDiBus is (leads the voltage in the BiDiBus, it would be best suited to the Master).

From an external power source (recommended)
Suitable for all applications and necessary if the GBMboost operated with booster function. The power supply (14V-20V DC) is connected to terminal X34 Pin1/Pin2.

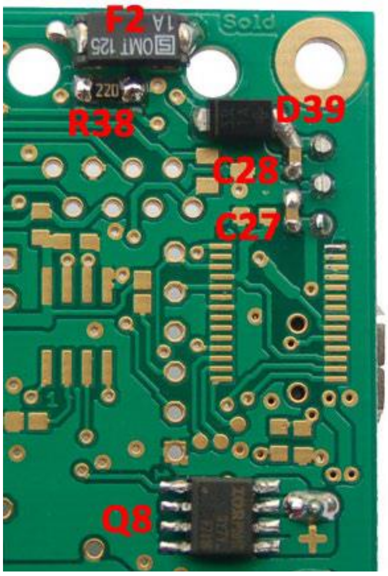
Conversion for the Master:

The GBMboost "master" must supply the BiDiBus with power (voltage)!

In this case, you have to build the diode D51 and replace it with a 500mA to 1A SMD fuse F2 (right panel).



An easier solution is the included resetting fuse F2 (SMD kit included) in parallel with the diode D51 to solder and flip over.
The left photo shows the parallel backup (round yellow component) to the diode on the master GBMboost.



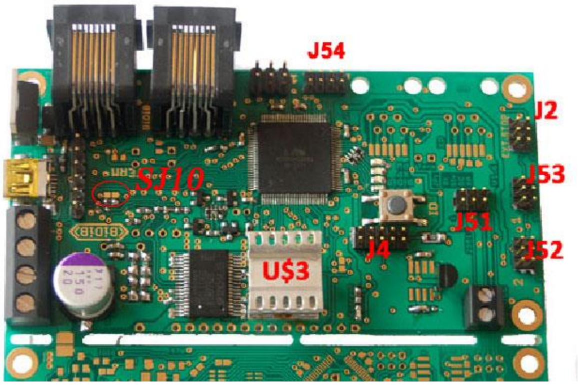
This conversion is only necessary on GBMboost master.

For GBMboost node remains the diode D51 equipped.

GBMboost the "Master" provides the BiDiBus with a 12V DC power via the F2 fuse. This leads to a larger line drop at 12V voltage regulator.

Therefore, the voltage regulator on GBMboost "master" can be equipped with the included heatsink.

By BiDiBus supply modules also receive BiDiB your power supply. Here one should keep in mind the actual consumption. It is recommended to each module using a separate supply voltage to increase in operation in order to avoid overloading the BiDiBus. The need for their node GBMboost booster function only one external supply there is an activation of the booster!



⚠

On **miniUSB** exclusively and a maximum of up to 5 Volt DC, or smoke from the USB UART FT 235RL.

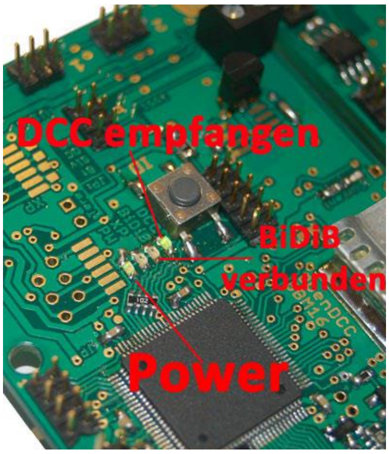
Depending on the application, some pin headers are omitted.
Short explanation for what application you need the individual headers:

J54	system settings	(required)
J2	external emergency stop button	(optional)
J4	Booster Display	(optional)
J51	connection to GBM16T (1)	
J53	connection to GBM16T (2)	
J52	connection to GBM16T (3)	
SJ10	DCC distribution, mode of operation	(depending on the options open in this case Option 1)

After the construction, the board again thoroughly with alcohol of all solder residue on both sides. Using the steps again carefully examine the entire structure. **Are you OK? -** Then it goes on.

For control via current-limited power supply to create the operating voltage at X34.
(Polarity / lettering on the back of platinum).

A master device is ready for use!
When a node device still requires the latest firmware using the wizard tool BiDiB be transferred. (see Chapter 5 "firmware update to the bootloader on node").



Empfangen = received
Verbunden = connected

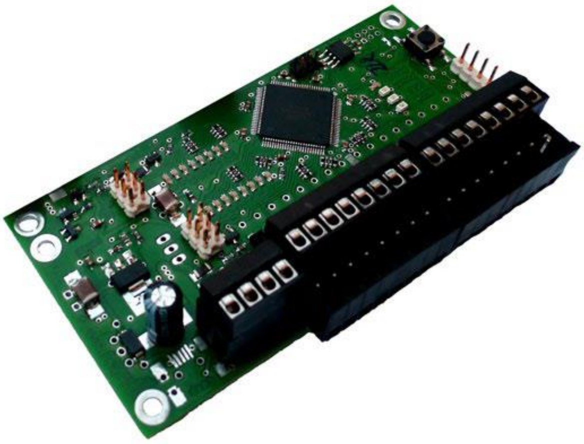
Normally now shimmers green **power LED** and the **DCC LED** indicates that a DCC signal is received. Each time the received and transmitted-BiDiB package **BiDiB LED** flickers.
The current rises to normal now 60mA - at 70mA.

You have now taken the GBMboost successfully.

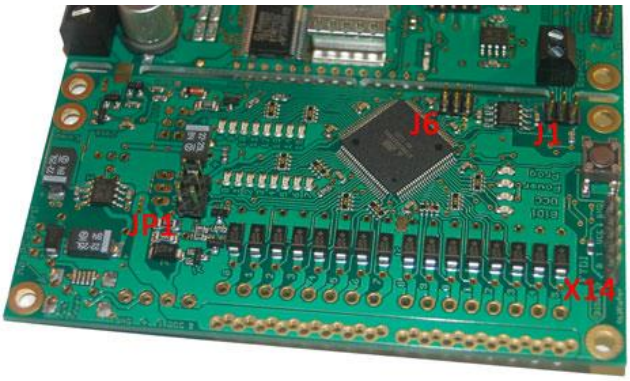
Status display on GBMboost

	State of the LED	remark
● DCC	DCC LED flickers	DCC signal is generated (the master) DCC signal comes through BiDiB (in Node)
	DCC from LED	No DCC signal fitting
● BIDIB	BiDiB LED lights (only when connected)	BiDiB ready for communication (the master) Connected to the BiDiBus (in Node)
	BiDiB flashes briefly (in operation)	It is a data communication instead of (access to the block)
	BiDiB LED off	BiDiB currently no communication / not connected to BiDiB
● XP	XP LED off	normal status
	XP LED on	Boot loader
● PW	POWER LED flickers	GBMboost is supplied with operating voltage
	Power LED flashes (blinks)	Identification started (by button or software function)
	DCC, BiDiB, XP and power LED flash	no eeprom file exists
	XP Power LED flash	no serial number available

3 Chapter: Equipping the GBM16T

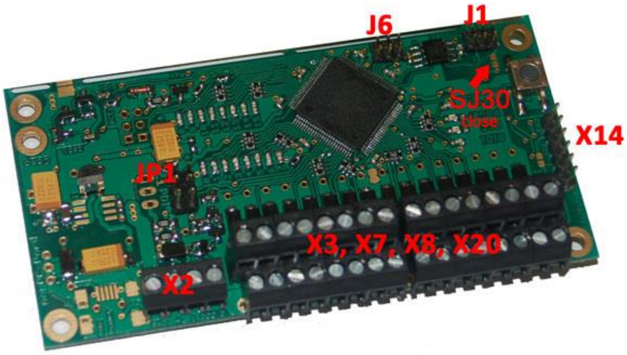


On the second part of the board is with its 16 GBM16T occupancy channels. The GBMboost can have two additional GMB16T modules are connected (total of three modules GBM16T).



THT solder components:

In the bag you will find enclosed the remaining parts to be assembled.
J1 ... PDI programming
J6 ... Settings of jumpers
J1 ... Connection to GBMboost (only necessary if the GBM16T is not connected to the web platinum / disconnected from GBMboost)
J9 ... Debug interface for firmware update (required if a firmware update to be performed without PDI Programmer / struck up bootloader is a prerequisite)



Almost there!
Only the terminal **X2, X3, X7, X8** and **X20**, which look different depending on the terminal!

J1 is to equip only if the module is not connected GBM16T over the web with a GBMboost. The connection is in this case made with a ribbon cable for GBMboost.

⚠ **SJ30 is close to or at the SMD-equipped version here a 22 ohm resistor** is built so that is signaled to the GBMboost that a detector is connected. With an open jumper or without the 22 ohm resistor on **GBM16T GBMboost** is not recognized.

Depending on requirements for the terminals, different types can be chosen. More information can be found on the website www.fichtelbahn.de.

Note when used with a loop module:

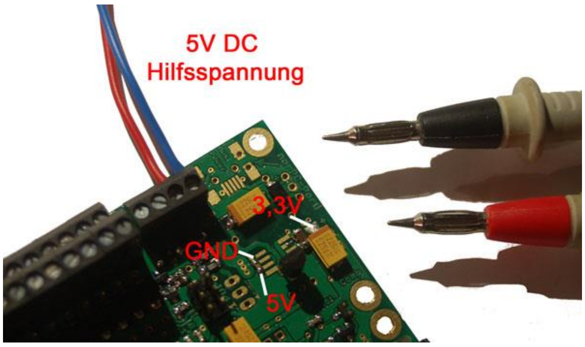
X2 is the terminal to use RIA AKL 059-04. Only then can the loop module is plugged into the motherboard. At higher terminals is for the terminal module can only be sent with a ribbon cable!

Depending on the application, some pin headers are omitted.
Short explanation for what application you need the individual headers:

J6	system settings	(required)
J1	to GBMboost communication	(optional)
JP1	PDI interface	(required)
SJ30	**GBM16T recognition**	(closed or fitted with 22R)
X14	FTDI interface for firmware update	(optional)
X2	5V DC (replacement feeding) and DCC track power	(required)
X3, X7, X8, X20	16 track outputs to ground	(required)

Control:
The **GBM16T** has two power supplies, the conventional supply from the DCC signal and an auxiliary voltage / power supply replacement for the "**occupancy report also booster failure**".

To check the correct functioning of **X2** (Pin1/Pin2) has a 5V voltage (replacement power supply) can be created with current limit.
On the component **C41** has a voltage of **5V** is measurable and the tantalum capacitor **C9** CPU Voltage **3.3 V**.



Hilfsspannung = auxiliary voltage

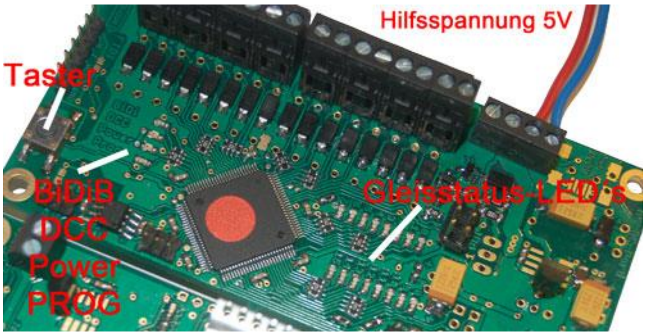
Important:
The auxiliary voltage / supply of spare GBM16T shall not come from the same ground source as the -GBM16T. i.e.

The GBMboost and GBM16T must be powered from two separate power supplies, otherwise there is a ground fault / short circuit!

When you add the voltage, the green **power LED** flickers and the **DCC LED** signaled by a flash that no DCC signal is received.

GBM16T makes the startup a self-test, which can be seen by the short duration of the track status LEDs.

The current rises to normal now 50mA - at 60mA.



Taster = button
Hilfsspannung = auxiliary voltage
Gleisstatus = track status

You have now successfully built the GBM16T.

Status display on GBM16T

	State of the LED	Remark
● BiDiB	BiDiB LED off	DCC signal without RailCom Cutout
	BiDiB LED flickers	DCC signal with RailCom Cutout
● DCC	DCC LED flashes (blinks)	No DCC signal on the track only auxiliary voltage
	DCC LED flickers	The tracks are supplied with DCC
	DCC LED lights (on)	Apply operating voltage with pressed button leads to the bootloader. Here, as long as the button is pressed, the DCC LED lights. Thereafter, only the Power LED!
● Power	POWER LED flickers	GBM16T is supplied with operating voltage
	POWER LED lights (on)	Bootloader active after starting with pressed key
● Prog	BiDiB, DCC, and Prog Power LED flash (blink)	no eeprom file exists
● Track	Track LED flickers	Decoder sends RailCom information
	Track LED lights (on)	Decoder does not send any information RailCom

4 Chapter: Commissioning the OpenDCC GBM

The GBMboost is delivered as ordered finished as a **master** or **node**. This means that the appropriate firmware flashed already on the block. The same is true for the GBM16T, this is already recorded with the current firmware.

A master device is ready for use!
When a node device still requires the latest firmware using the wizard tool BiDiB be transferred. (see chapter 5.2.2 "firmware update to the bootloader on node").

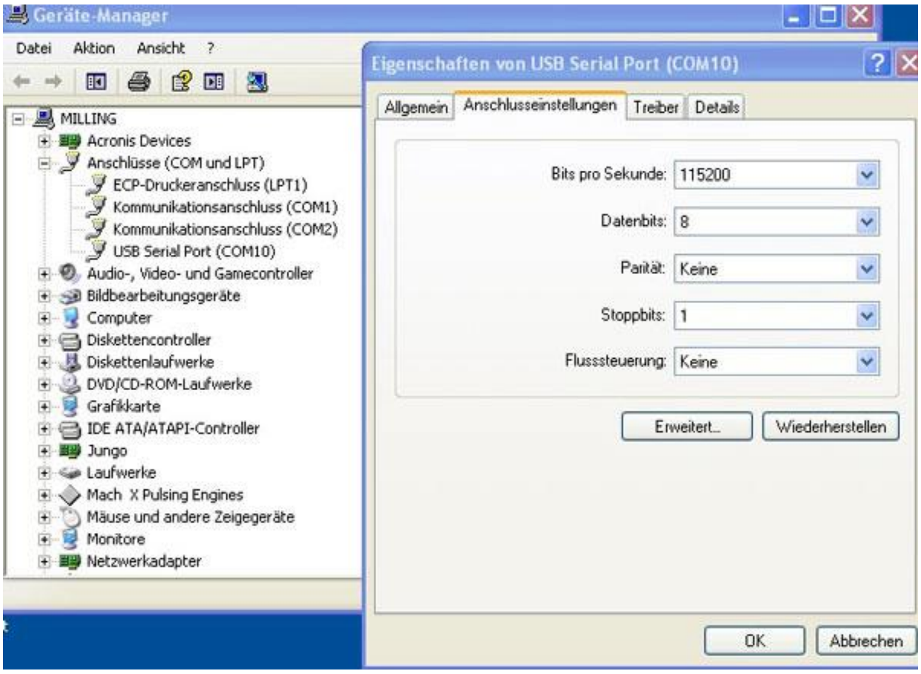
4.1 GBMboost only as an interface for BiDiB modules

Following the successful commissioning of the GBMboost the block can be connected to the PC via USB. This action and the master firmware will be played for the master device on GBMboost BiDiBus.

⚠ It may only appear on BiDiBus master device.

All other modules are GBMboost nodes (slave devices), and require the node firmware. These nodes are wired in series with an RJ45 cable with the master GBMboost.

⚠ A node cannot be connected to the PC via a USB cable!



Bits pro Sekunde: 115200
Datenbits: 8
Parität: Keine
Stoppbits: 1
Flusssteuerung: Keine

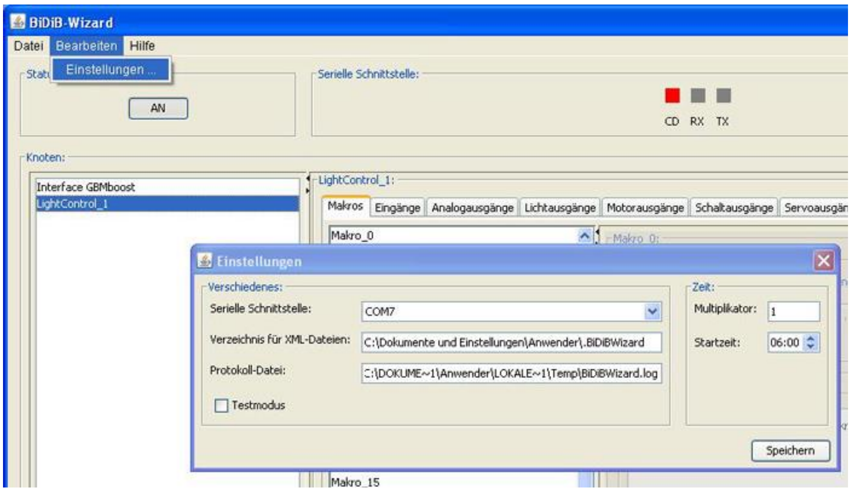
The PC will detect a virtual com port and installed this a standard driver.
This emulates a serial port on the PC, which can then access the control software. These VCP driver (if Windows has a driver) is found in www.ftdichip.com.

The actual traffic is made via the USB connection.

Alternative:
For virtual com port, there is also a special FTDI driver with manufacturer and product identifier. (see Configuring the FTDI USB chip).

The virtual COM port is entered as an additional COM port (eg COM10 or COM7) the application is then adjusted to this COM port.

The example of **BiDiB Wizard tool** in the settings of the program, the appropriate COM port are selected.



Serielle Schnittstelle: COM7

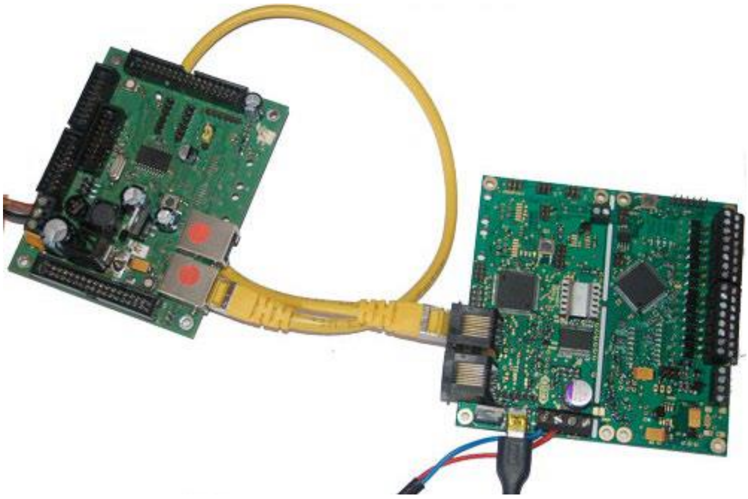
Verzeichnis für XML-Dateien: C:\Dokumente und Einstellungen\Anwender\.BiDiBWizard

Protokoll-Datei: :\DOKUME~1\Anwender\LOKALE~1\Temp\BiDiBWizard.log

☐ Testmodus

Multiplikator: 1

Startzeit: 06:00



The **GBMboost** is now ready for a connection to a BiDiB assembly.

If you connect a BiDiB module with the GBMboost on to match your BiDiB modules and on the GBMboost BiDiB illuminates the status LED.

The figure shows the **light control** on the **GBMboost**. The successful connection on the **BiDiBus** also appears in the list of nodes in **BiDiB Wizard tool** in the upper figure.

4.2 GBMboost as occupancy detector with GBM16T (Option 1)

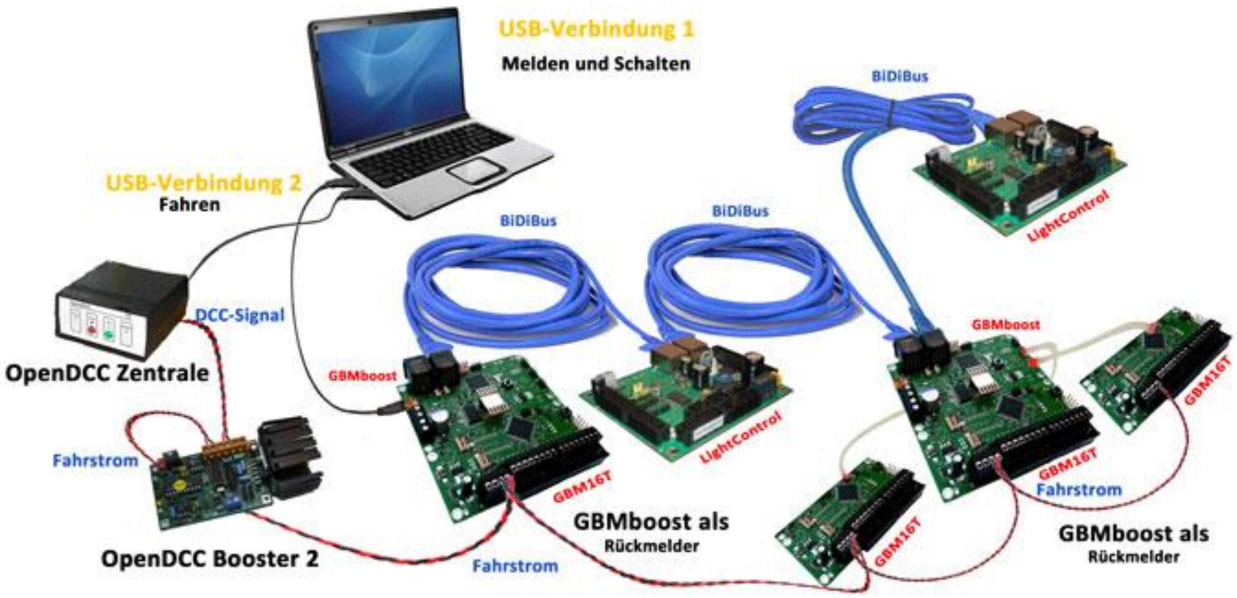
4.2.1 The concept:

In this first **Option 1** (**GBMboost** as occupancy detector) there is a master **GBMboost** as interface and also as occupancy detector for max. GBM16T three connected modules. All other **GBMboost** be the **node** (slave) and are only occupancy detector for their GBM16T modules (track sensor).

The only difference in the GBMboosts in this Option 1, the master needs the master firmware and the nodes need to be flashed with the node firmware.

A Command Station with built-in booster or as seen in the picture, a OpenDCC Z1 with separate booster generates the DCC speed commands and high-current.

If you here a RailCom - suitable headquarters and booster used, you get a feedback RailCom - Information (locomotive, heading ...) of a moving object. When not RailCom - enabled peripherals obtain the classical occupancy feedback!



Verbindung = connection, melden = report, schalten = switch, fahren = drive, fahrstrom = driving current, als rückmelder = as back detector

The jumper SJ5 and SJ6 need in this Option 1 (occupancy detector) remain open and may **not** be closed. **That would lead to a short circuit!**

Note:
When connecting other **modules** on a **GBM16T** GBMboost master node or they are only detected after the restart of the GBMboost. The TTL connection between GBMboost and GBm16T is **not** hot-plug capable.

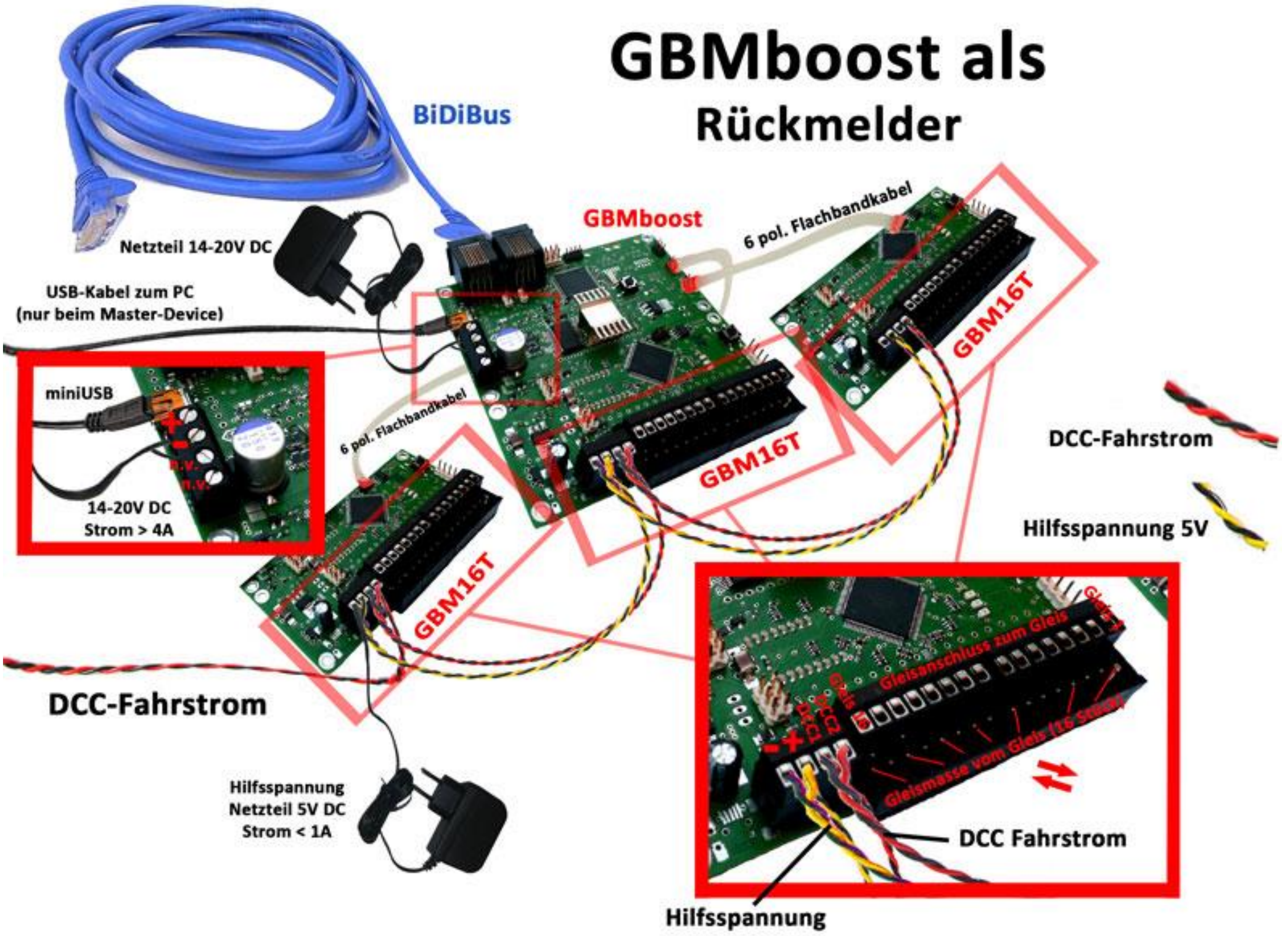
4.2.2 How the GBMboost GBM16T and connects!

The picture below shows the complete wiring of the now OpenDCC GBMs in Option 1:

Power supply:
The supply of **GBMboost** and **GBM16T** may not originate from the same ground reference, ie you will need two separate power supplies.

The **GBMboost** requires a power supply of 14V-20V to an output line of at least 4A. But you can feed multiple GBMboost modules together, but the output line total should be kept in mind.

The **GBM16T** can be supplied with an auxiliary voltage / spare power (5V DC 1A with sufficient). The replacement power can be built up in isolated operation. This means all GBM16T modules can be connected to the same GBMboost provide a 5V power supply.

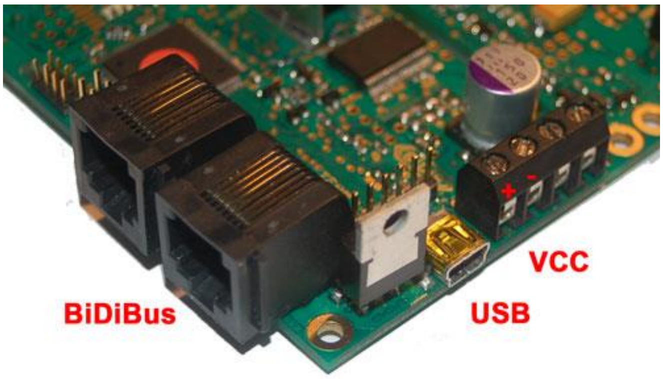


als rückmelder = as back detector, fahrstrom = driving current, netzteil = power supply, nur beim = only when, flachbandkabel = ribbon cable, hilfsspannung = auxiliary voltage, strom = current

Important:
Not all the GBM16T modules are powered from a 5V power supply!
Stand-alone operation, depending GBMboost note!!!

The GBMboost and GBM16T must be powered from two separate power supplies, otherwise there is a ground fault / short circuit!

Connectors on GBMboost:



Master:

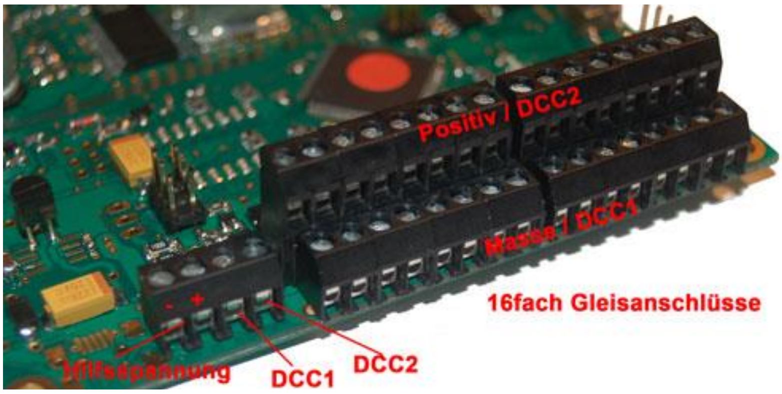
- **USB connection to PC** (*USB mini - jack*)

Master und Node:

- **All BiDiBus - modules are connected in series via the RJ45 jacks** (*the master GBMboost can also centrally, at the end and at the beginning of BiDiBus his place find*)

- **With external supply of terminal X34 Pin1 and Pin2 is used.** (*Note polarity marking on circuit backside*)

Connectors on GBM16T:



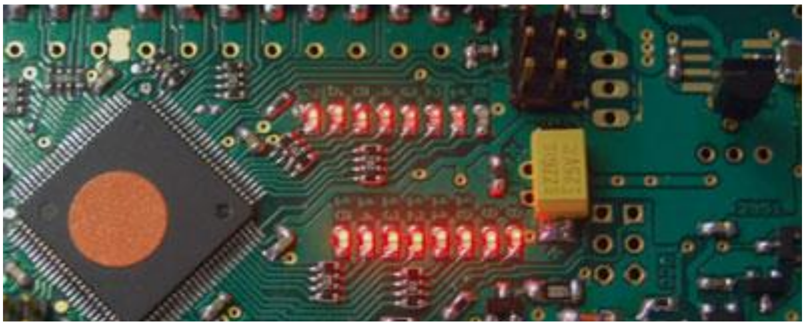
X2 pin1:	5V GND
X2 Pin2:	5V Plus
X2 Pin3:	DCC2
X2 Pin4:	DCC1
X3, X7, X8, X20:	16x

Sidings with or without mass

4.3 The OpenDCC GBM in operation

The **GBM16T** has a **DEMO mode** by simulating a busy message on all ports and clearing. This status is sent to the BiDiBus and can be read from a connected tool as occupancy report.

This is ideal for setting up and testing the assignments in PC software.



The **GBM16T** signaled locally occupied its current state using the 16 LEDs.

In Demo mode, you get a flash here shown on all detectors.

For more documentation see "**OpenDCC GBM in Action**" deals with the application page / application:

- GBMboost and GBM16T in BiDiB Wizard Tool
- Setting up and GBMboost GBM16T in rocrail

- Setting up and GBMboost GBM16T in Win-Digipet
- Read the booster status

5 Chapter: Firmware Update V1.6 GBMboost

This section is included as a supplement and explains how to perform a firmware update on GBMboost! This is helpful to the existing GBMboost a new development / adjust firmware!

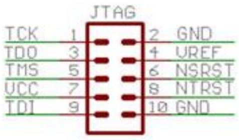
Two programming methods lead to the goal:
Direct programming i.e. Importing the firmware into the EEPROM and the flash with the help of a programmer, so the *.hex and *.eep -.. Files or via the easy bootloader principle.

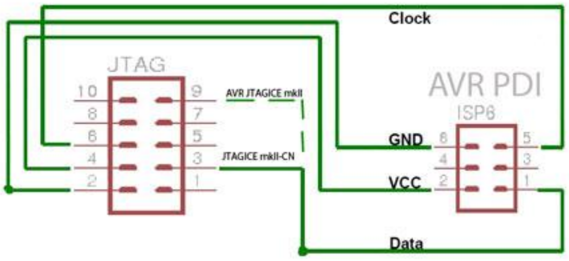
5.1 Programming the microcontroller via the Programmer

The ATXmega is recorded by means of PDI, which is a two-wire interface. The conventional SPI adapter (eg ponyprog) cannot be used.

Can be used:

- **AVRISPmkII**: This is an updated version of AVR Studio is required before using the programming menu sure to update the firmware of the AVRISP.
- **STK600**: In STK600 is drawn from the 6-pin connector blue PDI a 1:1 connection to the board. Important: On the STK600 must VTARGET jumpers are open!
- **JTAGICE mkII and JTAGICE mkII-CN:** The AVR JTAGICE mkII to connect the data (PDI) with the JTAG pin 9. Wherein JTAGICE mkII-CN (clone) the data (PDI) to be connected to the JTAG pin 3.



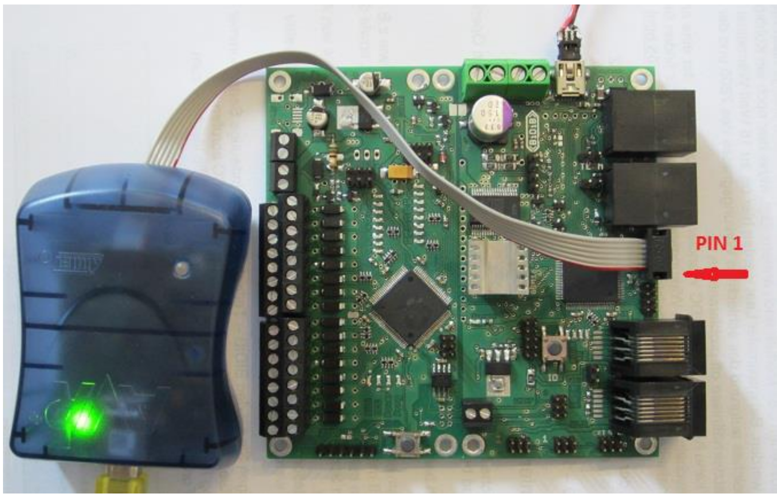


Note on the power supply during programming with the PDI adapter:

The GBMboost need to flash its own power supply and is not powered via the Programmer.

The following explanation and screenshots from the programming course reflect the version 4 of AVR Studio.

Connect the GBMboost with the power supply. To correct polarity and current limit set to 100 mA. Unprogrammed, the power consumption must not exceed 30mA.
The programming adapter with the PDI - input **J50** connect necessarily ensure correct positioning of PIN 1!



OpenDCC

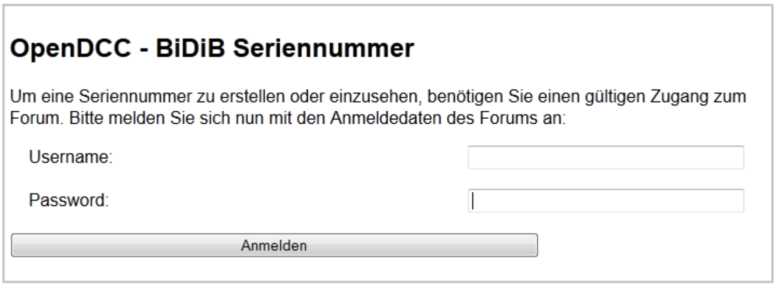
PIN 1 of the programming cable is shown by the red wire.

The latest software is available online for download from OpenDCC.
Download the required versions depending on the intended use:

- Master SW to operate as master / interface
- Node-SW (node) to operate as a node without an interface function
- The boot loader (the same for the master and node)
- The OpenDCC - BiDiB serial number

These files are packed. With a common archiver such as WinRar or Winzip unzip and best save in a separate folder.

The firmware:
For all modules, which support BiDiB, the assignment of a unique product identifier is mandatory based on a serial number.
This serial number can be generated on the following page without much effort:



OpenDCC - BiDiB Seriennummer

Um eine Seriennummer zu erstellen oder einzusehen, benötigen Sie einen gültigen Zugang zum Forum. Bitte melden Sie sich nun mit den Anmeldedaten des Forums an:

Username:

Password:

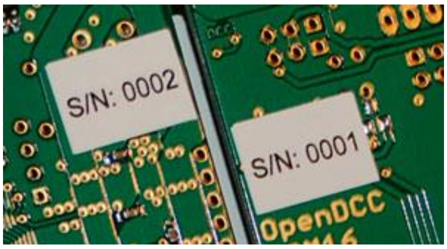
http://www.opendcc.de/elektronik/bidib/opendcc_bidib.html

To use the generator, registration is required. Here, the user ID and password from OpenDCC-forum (www.picotronic.ch/opendecc) is to be used.

How does it work?

- In the download files (firmware) to OpenDCC-BiDiB projects the VID and PID is already integrated, but no serial number is included. The download files can therefore be easily loaded onto the processor, any other existing serial number is not overwritten.
- If the firmware is not loaded at startup detects serial number, flashing an error code and the operation is blocked. Without a serial number is not the firmware!
- For each module must be **unique** to the generator a serial number generated.
- **Note down the serial number generated and please also note on the module.**
- The generator is an Intel Hex file generated with the contents of the serial number. This is displayed as text and available for download.
- The serial number must be in the normal way (ie via programming device (eg AVRmkII) or via the boot loader with the Wizard tool BiDiB) are imported into the assembly, other content will not be overwritten. This must be done only once.

About the serial number:



The serial number pasted on the back of the module is a hardware serial number and has nothing to do with the unique ID serial number for BiDiB modules.
Here are two different serial numbers.

The Bootloader:

The boot loader is part of the firmware, and cannot be omitted. The GBM but also has a dual boot loader (FTDI or BIDIB). This is activated when, during the starting of the module GBMboost the button is pressed and held. The firmware starts the boot loader with the jumper **J54, position J3** determines which interface will use:

- J3 plugged boat on the FTDI interface:
  Here is the same interface as for the debug interface using, but the baud rate is set to 19200 (8N1) The boat on the FTDI interface is activated when one before turning the GBM expresses the programming button and hold down. If you press again detaches, the bootloader responds with 'V GBM_Bootloader?.??' the FTDI interface. Now you can send a command to the boot loader, each entry with <cr> (Enter key) is completed.

  ... necessary for the firmware update on the master, more Informationenim point 4.3

- **J3 open** boot across the **BiDiBus interface**:
  The bootloader automatically logs on to BiDiBus and expect a firmware update. This can then be carried out via the BiDiBus interface with the **BiDiB wizard tool**.

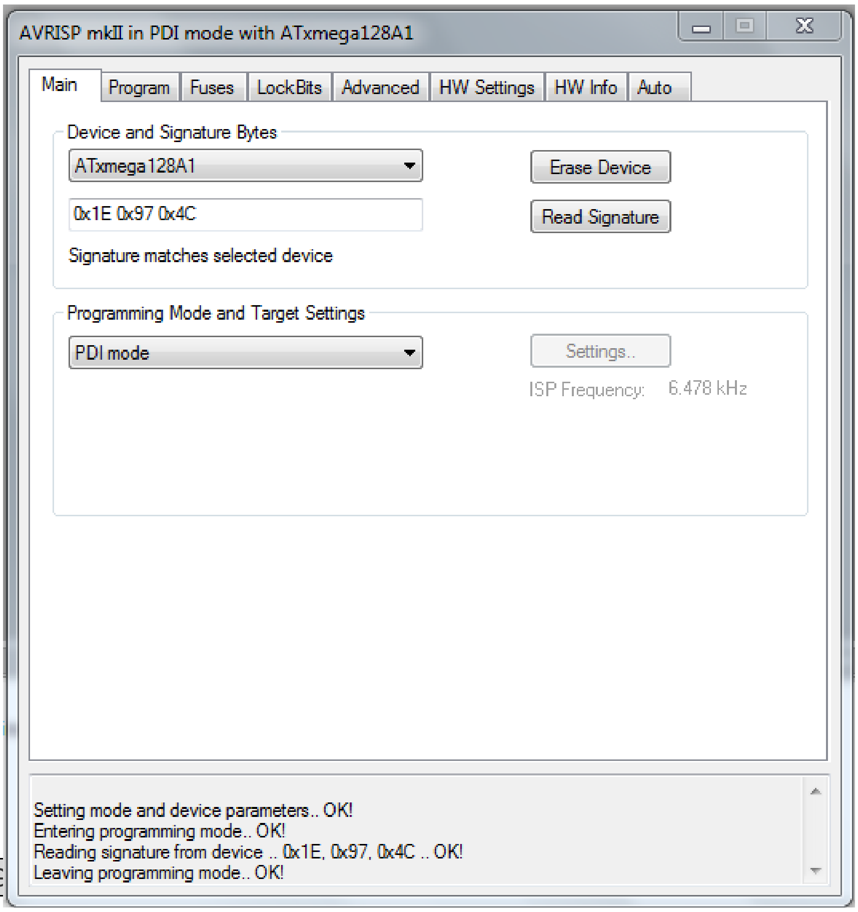
  ... necessary for the firmware update on the NODE, more information at the point 4

5.1.1 Programming with AVR Studio:

Start AVR Studio and build a connection. The screen would then after a successful login and connect via USB like this:

Select ATXmega128A1 and read from the signature for ATXmega128: 0x1E0x 97 0x4C.

Programming mode must be selected PDI.



AVRISP mkII in PDI mode with ATxmega128A1

Device and Signature Bytes: ATxmega128A1

0x1E 0x97 0x4C

Signature matches selected device

Programming Mode and Target Settings: PDI mode

ISP Frequency: 6.478 kHz

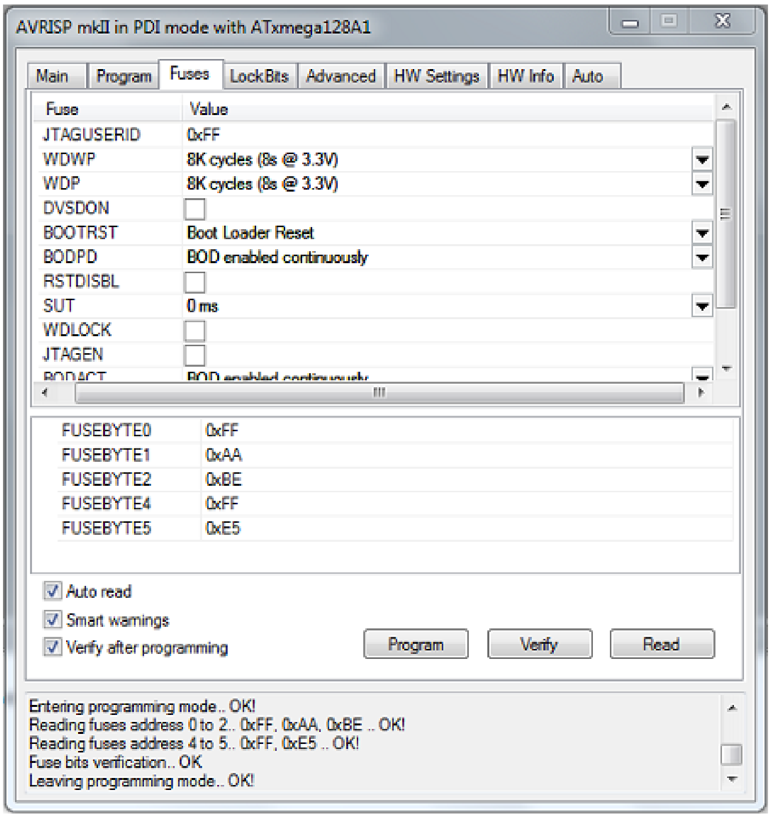
```
Setting mode and device parameters.. OK!
Entering programming mode.. OK!
Reading signature from device .. 0x1E, 0x97, 0x4C .. OK!
Leaving programming mode.. OK!
```

Next, the fuses can be set. To select the tab "FUSES". This is for setting the operating mode of the microcontroller. Go here very careful with the input! A "verfuster" Atmel can be difficult to revive.

Not visible settings:
BODLEVEL: auf 2.1 Volt
BODACT: BOD enabled continuously
EESAVE: Haken

In summary:
FuseByte 0: 0xFF
FuseByte 1: 0xAA
FuseByte 2: 0xBE
FuseByte 4: 0xFF
FuseByte 5: 0xE5

Note: the **EESAVE fuse** must be set. This prevents that with program updates and device resets the EEPROM is rewritten at the location of the serial number. The serial number is maintained until unchecking.
This device automatically **each** time we reset the EEPROM programming of the flash or the AVR Studio performed.



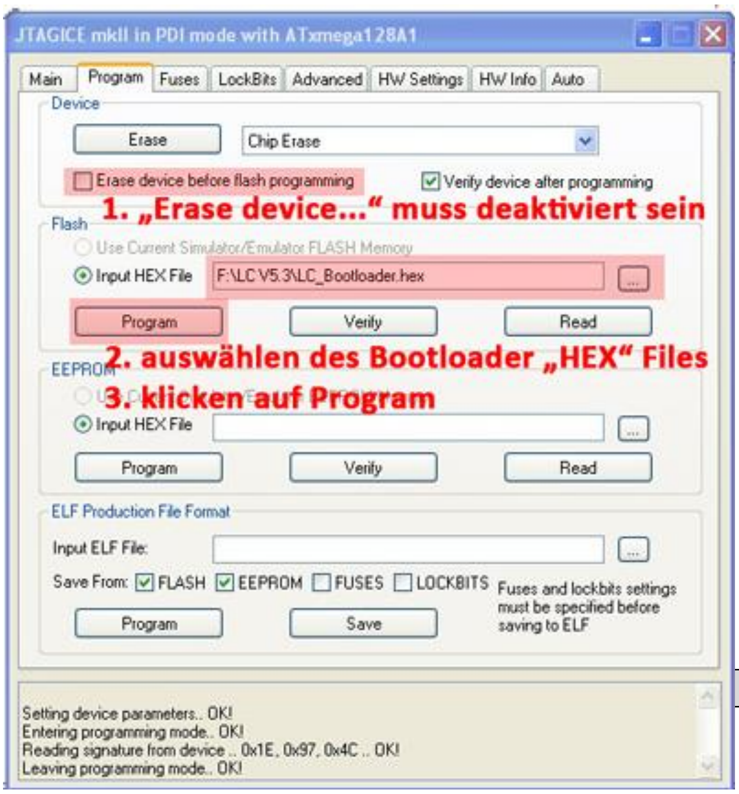
Fuse	Value
JTAGUSERID	0xFF
WDWP	8K cycles (8s @ 3.3V)
WDP	8K cycles (8s @ 3.3V)
DVSDON	□
BOOTRST	Boot Loader Reset
BODPD	BOD enabled continuously
RSTDISBL	□
SUT	0 ms
WDLOCK	□
JTAGEN	□
BODACT	BOD enabled continuously

FUSEBYTE0	0xFF
FUSEBYTE1	0xAA
FUSEBYTE2	0xBE
FUSEBYTE4	0xFF
FUSEBYTE5	0xE5

☑ Auto read
☑ Smart warnings
☑ Verify after programming

```
Entering programming mode.. OK!
Reading fuses address 0 to 2.. 0xFF, 0xAA, 0xBE .. OK!
Reading fuses address 4 to 5.. 0xFF, 0xE5 .. OK!
Fuse bits verification.. OK
Leaving programming mode.. OK!
```

Everything is set properly, AVR Studio reports OK!



Chip Erase

□ Erase device before flash programming ☑ Verify device after programming

1. „Erase device…“ muss deaktiviert sein

○ Use Current Simulator/Emulator FLASH Memory
● Input HEX File: F:\LC V5.3\LC_Bootloader.hex

2. auswählen des Bootloader „HEX“ Files

3. klicken auf Program

● Input HEX File

Input ELF File:
Save From: ☑ FLASH ☑ EEPROM □ FUSES □ LOCKBITS Fuses and lockbits settings must be specified before saving to ELF

```
Setting device parameters.. OK!
Entering programming mode.. OK!
Reading signature from device .. 0x1E, 0x97, 0x4C .. OK!
Leaving programming mode.. OK!
```

The box "**Erase flash device before programming**" must be deactivated otherwise will be deleted later when loading the firmware existing boot loader.
Now select the **bootloader HEX** file from your download folder and click on "**Program**" is the bootloader transferred to the GBMboost

possibly control:
To abort the programming process and restarting the GBMboost and subsequent **press on the button**, the boot loader. This does not, depending on the firmware and

remains JP 3 in boot mode or stand for BiDiB FTDI. The status is indicated by a lit LED.

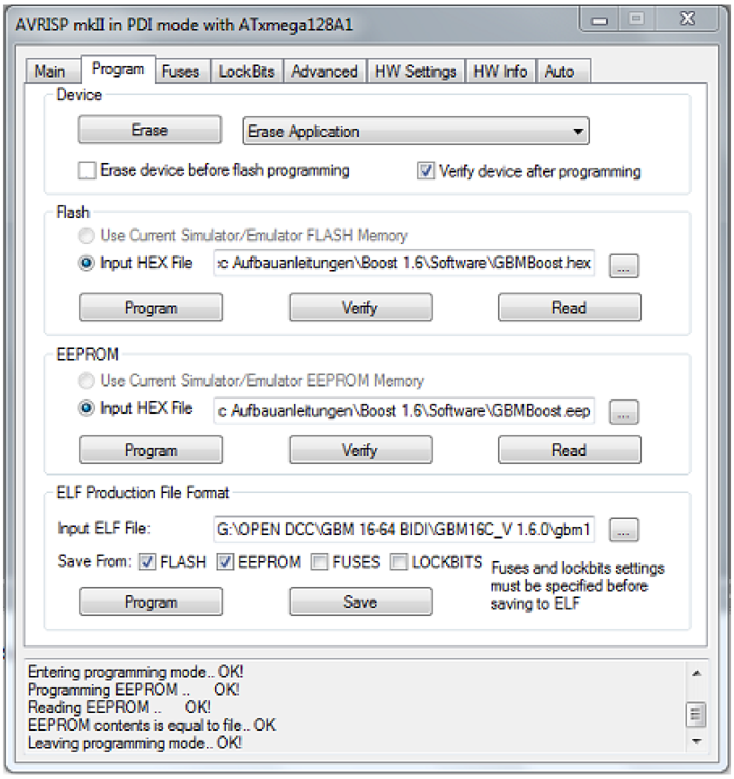
As a next step, we need the actual firmware, these are the two files * 000. Hex and * 001.hex.

The file * 000.hex belongs to the path "Flash". The file * 001.hex in the path "EEPROM".

⚠ The **GBMboost** can be used as master or node. The difference is in the firmware, so in this case the correct Master or Node firmware transfer.

By clicking on the firmware **program** is transferred to the microcontroller.

Do not forget to perform this procedure separately in **Flash** and **EEPROM**. The two files are not automatically transferred.



AVRISP mkII in PDI mode with ATxmega128A1

Device

Erase Application

□ Erase device before flash programming ☑ Verify device after programming

Flash

○ Use Current Simulator/Emulator FLASH Memory

● Input HEX File: :c Aufbauanleitungen\Boost 1.6\Software\GBMBoost.hex

EEPROM

○ Use Current Simulator/Emulator EEPROM Memory

● Input HEX File: c Aufbauanleitungen\Boost 1.6\Software\GBMBoost.eep

ELF Production File Format

Input ELF File: G:\OPEN DCC\GBM 16-64 BIDI\GBM16C_V 1.6.0\gbm1

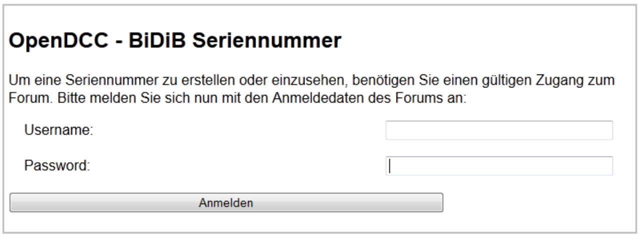
Save From: ☑ FLASH ☑ EEPROM □ FUSES □ LOCKBITS

Fuses and lockbits settings must be specified before saving to ELF

```
Entering programming mode.. OK!
Programming EEPROM ..     OK!
Reading EEPROM ..     OK!
EEPROM contents is equal to file.. OK
Leaving programming mode.. OK!
```

The last step is to generate a cost serial number with the serial number BIDIB tool to http://www.opendcc.de/elektronik/bidib/opendcc_bidib.html.

This registration in OpenDCC forum is necessary. By choosing the right product and a remark, the serial number file is generated. This file can be saved on your PC with a click on the colored symbol. Eep.



OpenDCC - BiDiB Seriennummer

Um eine Seriennummer zu erstellen oder einzusehen, benötigen Sie einen gültigen Zugang zum Forum. Bitte melden Sie sich nun mit den Anmeldedaten des Forums an:

Username:

Password:

The serial number is also a **eeprom file** or a file **serial_000.hex** the selected point in the EEPROM and must be transferred to the GBMboost by **clicking Program**.

Do not forget to note the serial number on the board.

Error messages:

The four green status LEDs blink now frantically

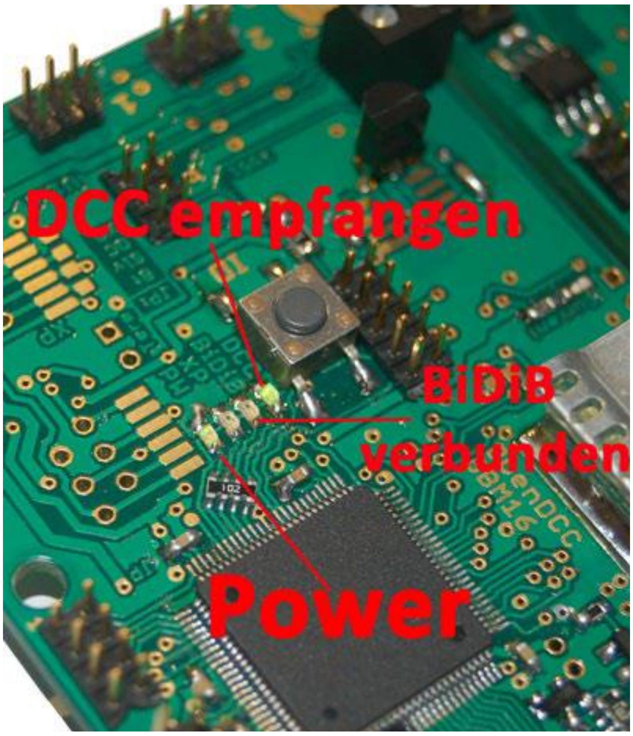
- The eeprom file was transferred to the Forgot GBMboost

The bottom two green status LEDs blink now frantically

- The serial number is not transferred to the GBMboost

Normally now shimmers green **power LED** and the **DCC LED** indicates that a DCC signal is received. Each time the received and transmitted-BiDiB package flickers the **BiDiB LED**.
The current rises to normal now 60mA - at 70mA.

empfangen = receive
verbunden = connected



You have now programmed the GBMBoost successfully.

5.2 Microcontroller programming via the boot loader

Note:
The update procedure is divided in Master and Node (Slave). Please note!

5.2.1 The master:

GBMboost as the master can only be updated through this procedure or put into service, unless the programmer - is chosen option.

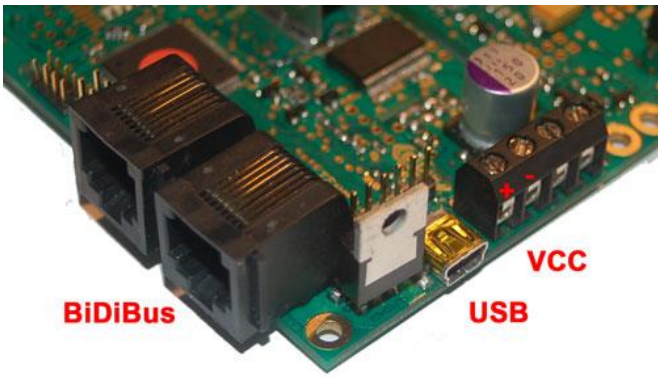
⚠ **The BiDiB Wizard tool to update the firmware or does not play for the first time when the Master!**

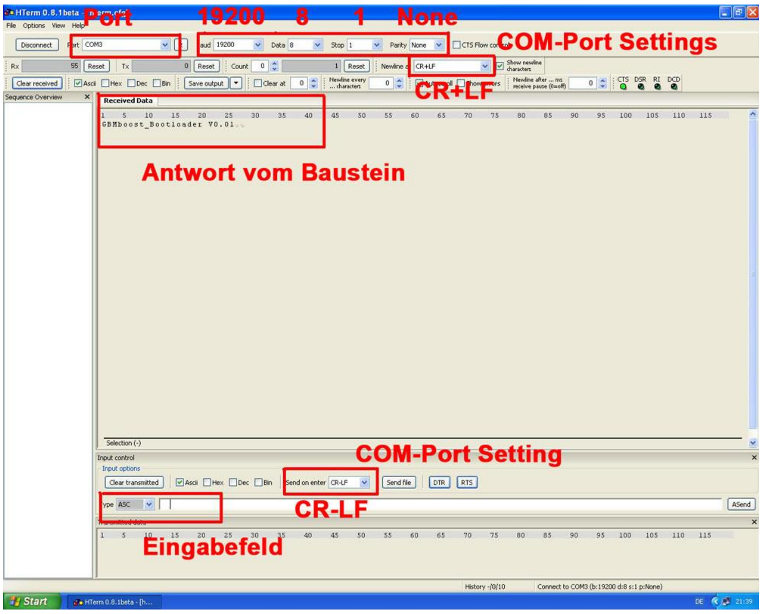
The prerequisite for this option is that the microcontroller has been flashed with a bootloader and the FUSES are set. The first steps of section 4.2.2 are required.

or ... you have already purchased a ATXmega-loaded with bootloader in Fichtel web shop. **In the latter case no programmer (section 4.2.2) is required.**

The **GBMboost Master** has a USB port that you can use to update the firmware using a terminal program.

This requires that the GBMboost be connected via USB to the PC.





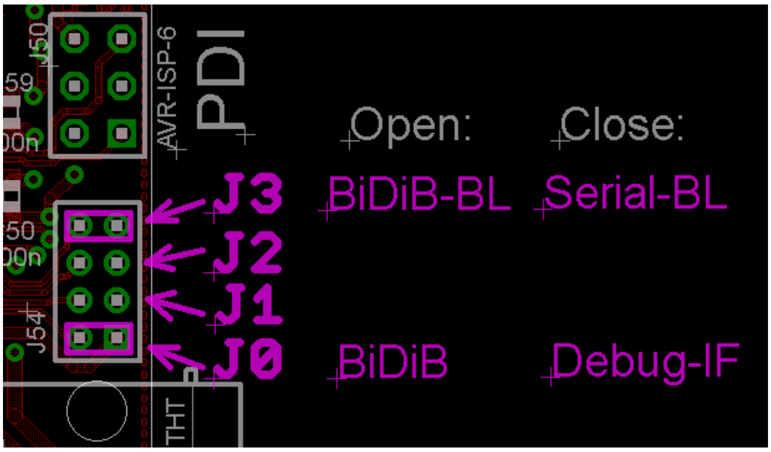
Port COM3, Baud 19200, Data 8, Stop 1, Parity None, Newline at CR+LF

```
GBMboost_Bootloader V0.01
```

Send on enter CR-LF

Connect to COM3 (b:19200 d:8 s:1 p:None)

In the device manager of your operating system is recognized as GBMboost new virtual COM interface. For communication with the GBMboost must COM port and the terminal program to **19200 baud and 8N1** operate.



The **GBMboost** has a bootloader for FTDI (serial) and BiDiB. By the set of jumper **J54, position J3** from FTDI (Serial) bootloader is selected. In the case of the Masters is the USB interface.

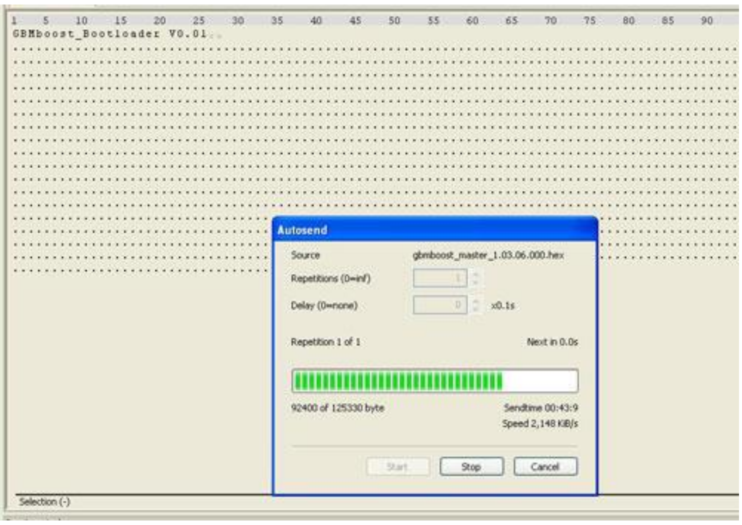
When a node receives the BiDiB not be updated and no FTDI chip is available (USB port), the debug interface can be used with an FTDI cable. So far jumper J0 must be set.

Hold down the **button** on the GBMboost and switch on the supply voltage for the GBMboost.

Now they make a call to the block by hterm on the **Connect** button in the terminal program click.

Now you send a "?" And confirm by pressing **Enter**.

The GBMboost answers "GBMboost_Bootloader V?"



```
GBMboost_Bootloader V0.01..
```

Autosend

Source: gbmboost_master_1.03.06.000.hex

Repetitions (0=inf): 1

Delay (0=none): 0 x0.1s

Repetition 1 of 1 — Next in 0.0s

92400 of 125330 byte — Sendtime 00:43:9 — Speed 2,148 KiB/s

Step1:
Now you send a f and press **Enter**. The entry is made in the field of the terminal program.

The GBMboost responds with a dot.

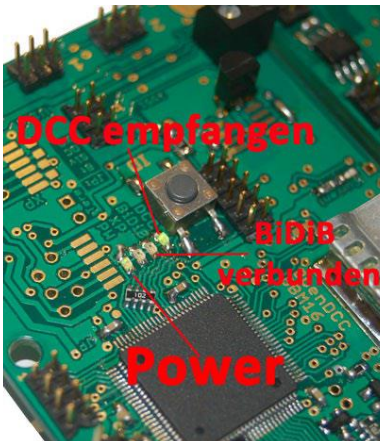
Step2:
Now click on the button "**Send file**" and choose the appropriate flash firmware file (* or * 000.hex hex), and press Start. The first part of the firmware is transmitted to the GBMboost, visible through the many points in the terminal program.

Step3:
Now you have to transfer the eeprom. To send an **e** and confirm with **Enter**. The GBMboost responds with a dot. Choose the appropriate eeprom file (*. Eeprom or 001.hex *) and press **Start**. There are a few points back.

Step4:
The last step is still the solid serial number is transmitted (see 4.2 serial number). To send an **e** again and press **Enter**. Now select the solid serial number and press Start.

Do not forget to note the serial number on the board.

The **GBMboost** is successfully updated or put into service. After disconnect the power supply, remove the jumper **J3** and re-connect the power supply, the **power LED** flickers.

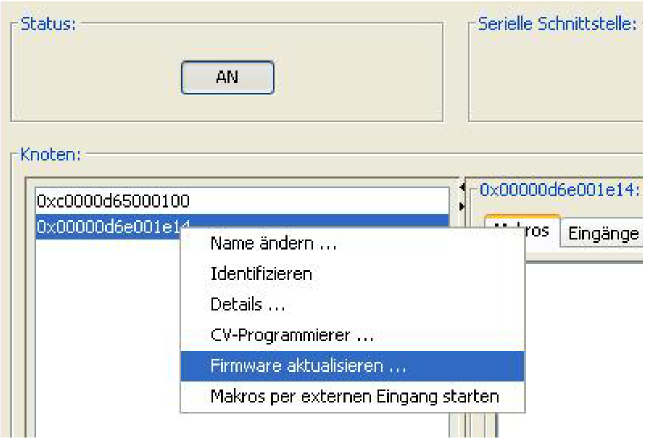


5.2.2 When Node:

Basically when **GBMboost Node** works the option described by GBMboost master (section 5.2.1), but the function of **firmware update** over the BiDiB Wizard tool significantly more user-friendly.

The advantage is:
The GBMboost can when mounted on the tool is updated, one also speaks of a remote service.

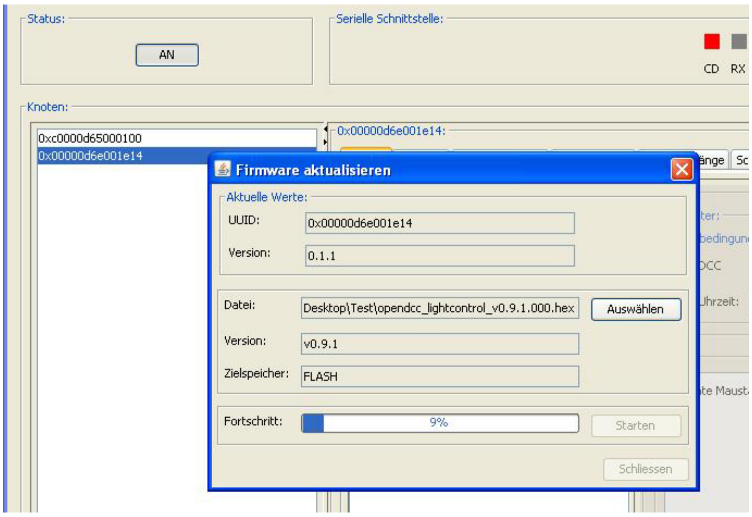
In the node list of the BiDiB Wizard tool, see the GBMboost BiDiB block and all other components listed for a configuration or a firmware update.



The necessary firmware and serial number is transmitted to the BiDiB Wizard tool. The aufzuspielenden firmware - files must be in the following notation:

gbmboost_node.hex	[FLASH]	-->	gbmboost_node_v?.?.?.000.hex
gbmboost_node.eep	[EEprom]	-->	gbmboost_node_v?.?.?.001.hex
_???fe????????.eep	[S/N]	-->	????fe????????_serial.001.hex

The GBMboost is applied to the supply voltage and connected to the BiDiBus. **The first time you flash all LEDs are off.**



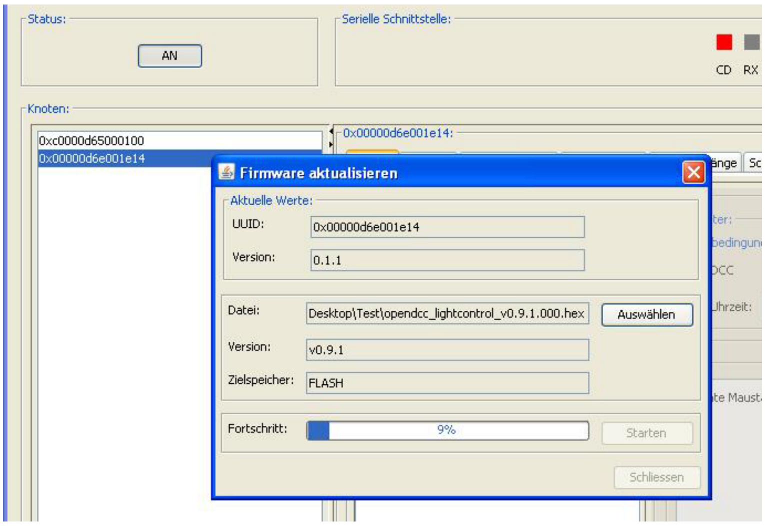
Aktuelle Werte:
UUID: 0x00000d6e001e14
Version: 0.1.1
Datei: Desktop\Test\opendcc_lightcontrol_v0.9.1.000.hex
Version: v0.9.1
Zielspeicher: FLASH
Fortschritt: 9%

The **BiDiB wizard tool** will open and in the node list, the serial number of the GBMboost Masters (interfaces) to be visible.

When pressing the button on the new GBMboost illuminates the second green LED and the tool displays a new number **... the new GBMboost has awakened**.

Right-click on the serial number of the new GBMboosts click and from the context menu action "**Update Firmware**" menu.

The next step is to select the folder with the three firmware files (see above).
These files must be in the order *** 000.hex**, ***001.hex** and ****_??fe?serial.001.hex** Transmitted.
Close the window with the GBMboost node is updated to the new firmware and ready.



Status:

AN

Serielle Schnittstelle:

Knoten:

0xc0000d65000100
0x00000d6e001e14

0x00000d6e001e14:

Firmware aktualisieren

Aktuelle Werte:

UUID: 0x00000d6e001e14

Version: 0.1.1

Datei: Desktop\Test\opendcc_lightcontrol_v0.9.1.000.hex

Version: v0.9.1

Zielspeicher: FLASH

Fortschritt: 9%

**** Serial number must be only partly filled, if no S / N GBMboost is available on the ... So only during initial startup and not the update.**

Do not forget to note the serial number on the board.

6 Chapter: Firmware update on GBM16T

6.1 Programming the microcontroller with the programmer

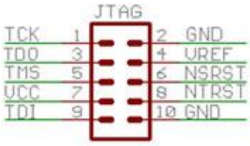
The ATXmega is recorded by means of PDI, which is a two-wire interface.
The conventional SPI adapter (eg ponyprog) cannot be used.
For GBM16T no serial number is required.

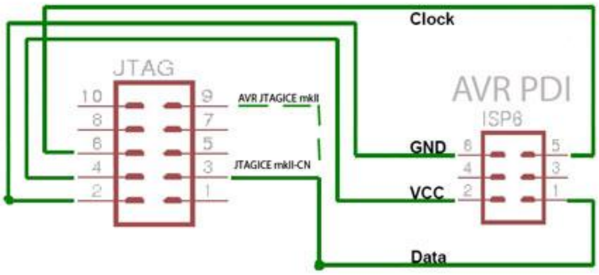
Can be used:

- **AVRISPmkII**: This is an updated version AVR Studio required before use necessarily in the programming menu, the firmware updating the AVRISP.

- **STK600**: When the 6-pin STK600 PDI blue plug on a 1:1 connection the board considered. Important: On the STK600 must open the VTARGET jumper!

- **JTAGICE mkII** and JTAGICE mkII-CN: In the AVR JTAGICE mkII is Data (PDI) with the JTAG pin to connect the 9th. In the JTAGICE mkII-CN (Clone) is the data (PDI) with the JTAG pins to connect the third





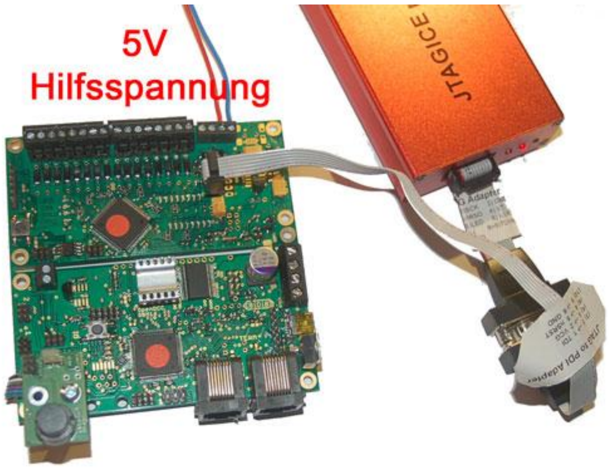
Note on the power supply during programming with the PDI adapter:
The GBM16T need to flash its own power supply and is not powered via the Programmer.

The following notes and screenshots from the programming course enter the version 4 of AVR Studio again.

Connect the **GBM16T** with the power supply. To correct polarity and current limit set to 100 mA.
Unprogrammed, the power consumption must not exceed 20mA.

Connect input **J1**, pay attention to correct positioning of PIN 1 - the programming adapter with the PDI!

PIN 1 of the programming cable is shown by the red wire.



Hilfsspannung = auxiliary voltage

Important Power Considerations in the programming process:
The programming adapter connects the GND of PC with the GBM16T.

Therefore, make sure that:

- **The GBMBoost / GBM16T is made completely ungrounded**
- **No USB connection to the PC is the GBMboost**

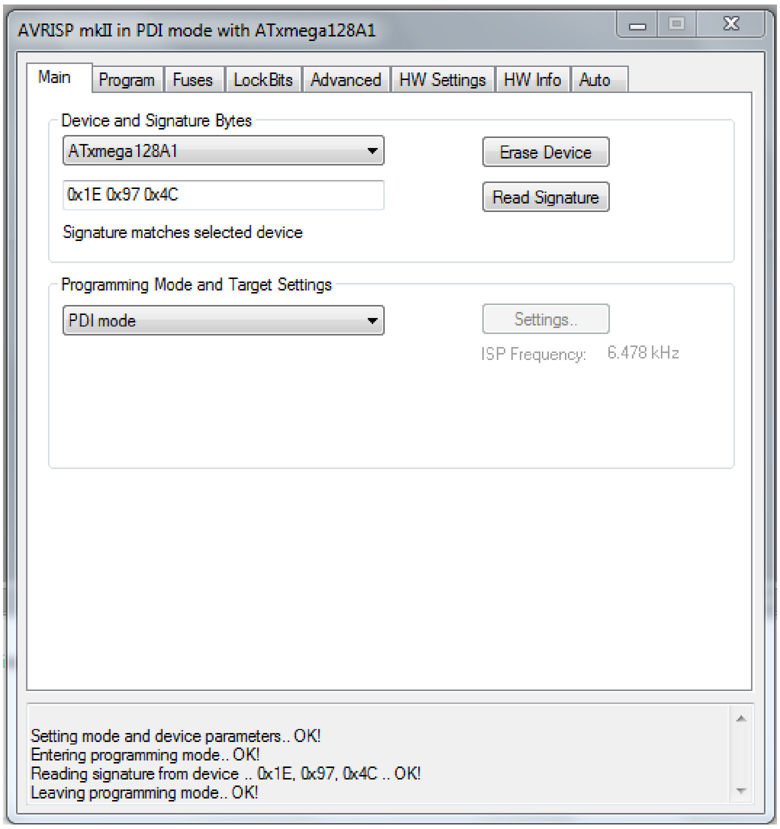
This is especially true when programming via the PDI port when the bridges **SJ5** and **SJ6** are closed at the initial programming already.
A later firmware update can be done safely on the BiDiBus, here is no action necessary.

Programming with AVR Studio:

Start AVR Studio and build a connection. The screen would then after a successful login and connect via USB like this:

Select ATXmega128A1 and read from the signature for ATXmega128: 0x1E0x 97 0x4C.

Programming mode must be selected PDI.



Device and Signature Bytes
ATxmega128A1
0x1E 0x97 0x4C
Signature matches selected device

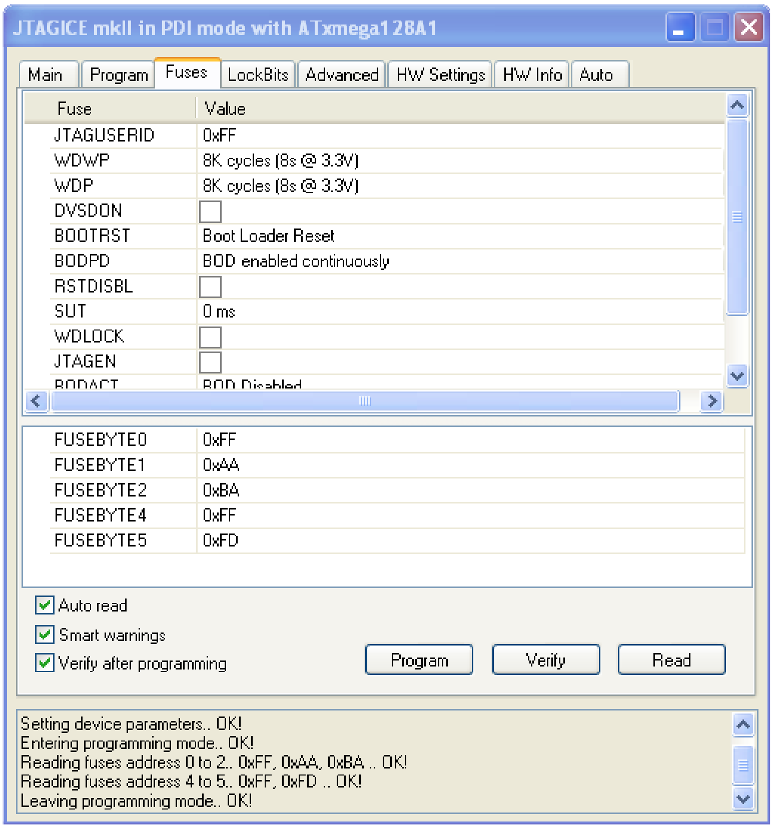
Programming Mode and Target Settings
PDI mode
ISP Frequency: 6.478 kHz

```
Setting mode and device parameters.. OK!
Entering programming mode.. OK!
Reading signature from device .. 0x1E, 0x97, 0x4C .. OK!
Leaving programming mode.. OK!
```

Next, the fuses can be set. To select the tab "FUSES". This is for setting the operating mode of the microcontroller. Go here very careful with the input! A "verfuster" Atmel can be difficult to revive.

Not visible settings:
BODLEVEL: to 2.1 volts
BODACT: BOD disable

In summary:
FuseByte 0: 0xFF
FuseByte 1: 0xAA
FuseByte 2: 0xBA
FuseByte 4: 0xFF
FuseByte 5: 0xFD


JTAGICE mkII in PDI mode with ATxmega128A1

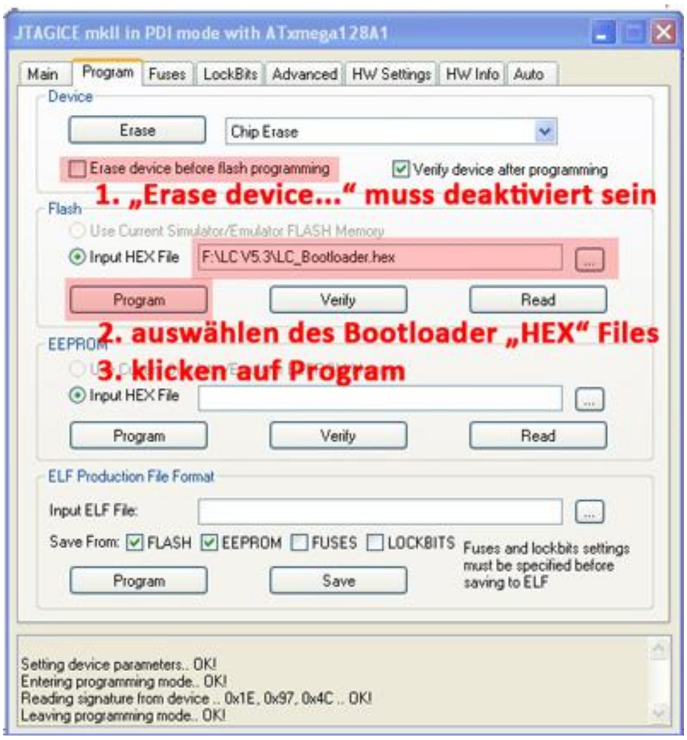
Fuse	Value
JTAGUSERID	0xFF
WDWP	8K cycles (8s @ 3.3V)
WDP	8K cycles (8s @ 3.3V)
DVSDON	□
BOOTRST	Boot Loader Reset
BODPD	BOD enabled continuously
RSTDISBL	□
SUT	0 ms
WDLOCK	□
JTAGEN	□
BODACT	BOD Disabled

FUSEBYTE0	0xFF
FUSEBYTE1	0xAA
FUSEBYTE2	0xBA
FUSEBYTE4	0xFF
FUSEBYTE5	0xFD

☑ Auto read
☑ Smart warnings
☑ Verify after programming

```
Setting device parameters.. OK!
Entering programming mode.. OK!
Reading fuses address 0 to 2.. 0xFF, 0xAA, 0xBA .. OK!
Reading fuses address 4 to 5.. 0xFF, 0xFD .. OK!
Leaving programming mode.. OK!
```

Everything is set properly, AVR Studio reports OK!


JTAGICE mkII in PDI mode with ATxmega128A1

Chip Erase
□ Erase device before flash programming
☑ Verify device after programming

1. „Erase device…“ muss deaktiviert sein

○ Use Current Simulator/Emulator FLASH Memory
● Input HEX File: F:\LC V5.3\LC_Bootloader.hex

2. auswählen des Bootloader „HEX“ Files

3. klicken auf Program

● Input HEX File

Input ELF File:
Save From: ☑ FLASH ☑ EEPROM □ FUSES □ LOCKBITS
Fuses and lockbits settings must be specified before saving to ELF

```
Setting device parameters.. OK!
Entering programming mode.. OK!
Reading signature from device .. 0x1E, 0x97, 0x4C .. OK!
Leaving programming mode.. OK!
```

The box "**Erase flash device before programming**" must be deactivated otherwise will be deleted later when loading the firmware existing boot loader.
Now select the file from your download folder **xboot_gbm16t.hex** and click on "**Program**" is the bootloader transferred to the GBM16T.

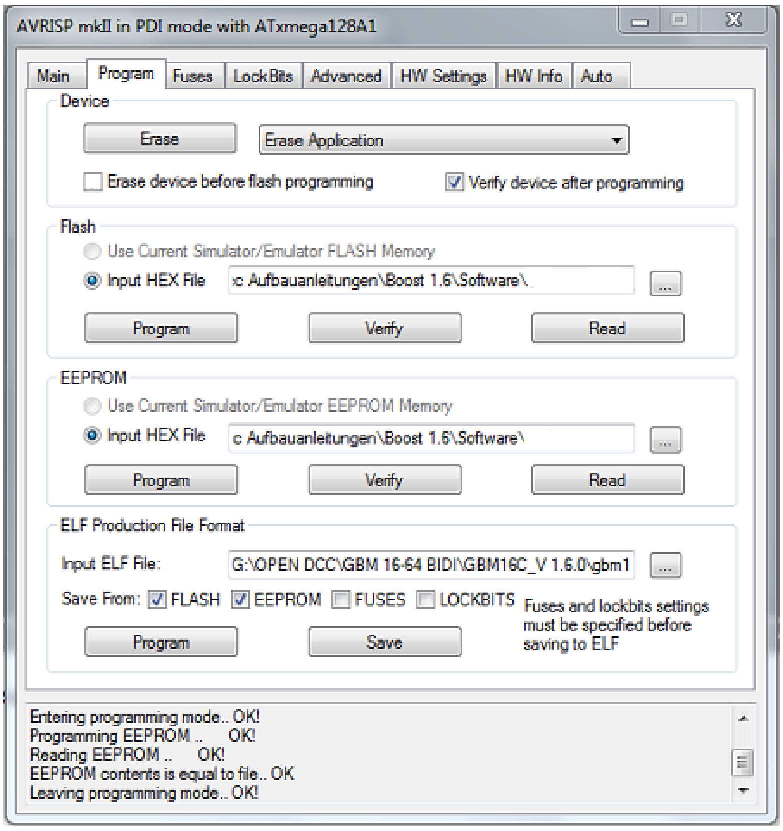
possibly control:

To abort the programming process and restarting the GBM16T and **subsequent press on the button**, the boot loader. This is no firmware in boot mode and is available for FTDI. The status is indicated by a lit LED.

As a next step, we need the firmware, these are the two files * gbm16t_. Gbm16t_ hex and *. Eep.

The file **gbm16t_*.hex** belongs to the path "Flash".
The file **gbm16t_*.eep** in the path "EEPROM".

By clicking on the firmware program is transferred to the microcontroller.
Do not forget to perform this procedure separately in Flash and EEPROM. The two files are not automatically transferred.



AVRISP mkII in PDI mode with ATxmega128A1

Device
Erase Application
□ Erase device before flash programming ☑ Verify device after programming

Flash
○ Use Current Simulator/Emulator FLASH Memory
● Input HEX File :c Aufbauanleitungen\Boost 1.6\Software\

EEPROM
○ Use Current Simulator/Emulator EEPROM Memory
● Input HEX File c Aufbauanleitungen\Boost 1.6\Software\

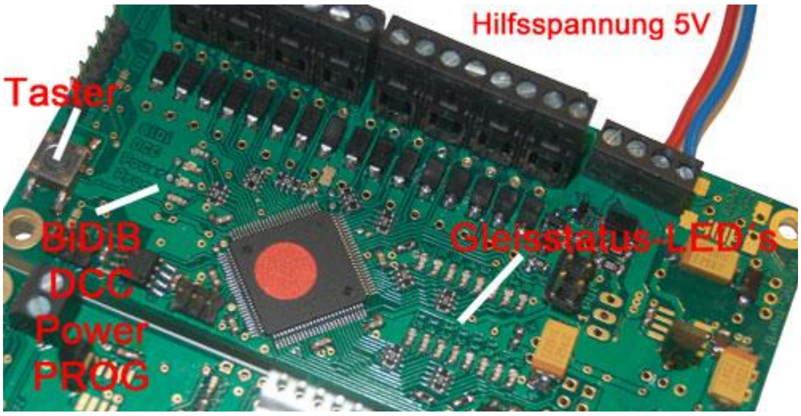
ELF Production File Format
Input ELF File: G:\OPEN DCC\GBM 16-64 BIDI\GBM16C_V 1.6.0\gbm1
Save From: ☑ FLASH ☑ EEPROM □ FUSES □ LOCKBITS
Fuses and lockbits settings must be specified before saving to ELF

```
Entering programming mode.. OK!
Programming EEPROM ..      OK!
Reading EEPROM ..     OK!
EEPROM contents is equal to file.. OK
Leaving programming mode.. OK!
```

Note:
There are a gbm16t_22ohms version and a version gbm16t_5R6ohms. The firmware is a function of the selected resistance **R103 to R119** placement. By 5R6 ohm version no substitute measurement (occupancy report also booster failure) is possible, this is the hardware RailCom compliant.

Error messages:
The four status LEDs blink now frantically

- The eeprom file was transferred to the Forgot GBMboost



After the flash now the green **power LED** and the **DCC LED** flickers signaled by a flashing that no DCC signal is received.

GBM16T makes the startup a self-test, which can be seen by the short duration of the light rail status LEDs.

The current rises to normal now at 50mA - 60mA.

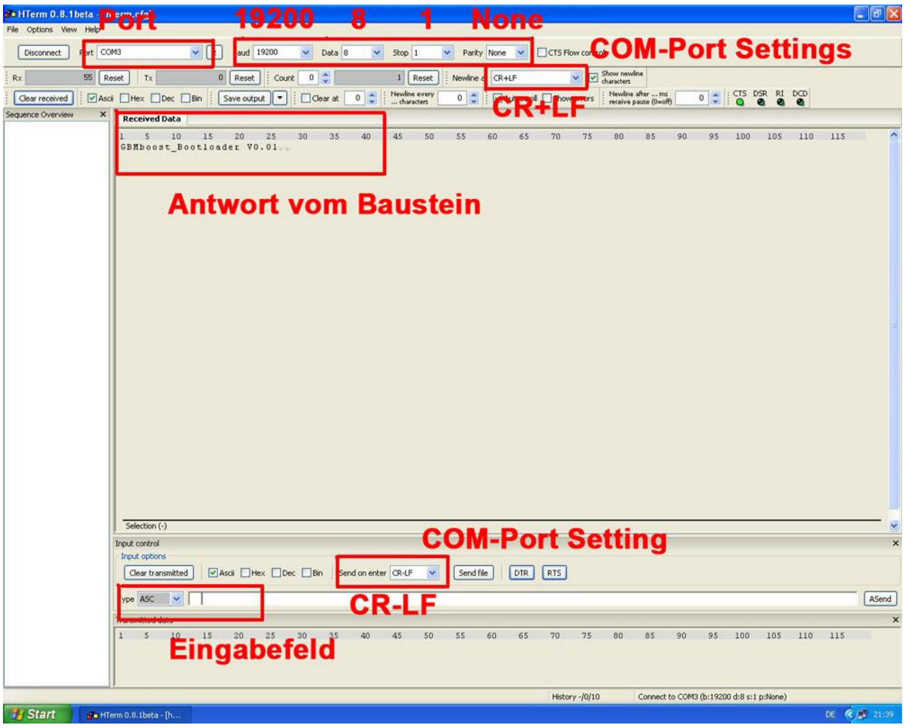
You have now programmed the GBM16T and successfully put into operation.

6.2 Programing the microcontroller with the Bootloader

The GBM16T can only be updated through this procedure or put into service, unless the programmer - is chosen option.

⚠ The BiDiB Wizard tool to update the firmware does not work for the master!

You need a FDTI RS232 TTL cable (cost 20 euros) and a terminal program (eg hterm).



HTerm 0.8.1beta

Port COM3 · Baud 19200 · Data 8 · Stop 1 · Parity None · Newline at CR+LF

Received Data:

```
GBMboost_Bootloader V0.01
```

Send on enter CR-LF · Type ASC

Connect to COM3 (b:19200 d:8 s:1 p:None)

In the device manager of your operating system, the TTL cable is detected as a new virtual COM interface. For communication with the GBM16T must COM port and the terminal program to **19200 baud and 8N1** to operate.

Now you make a call to the module by clicking on the **Connect** button in the terminal program "hterm" button.

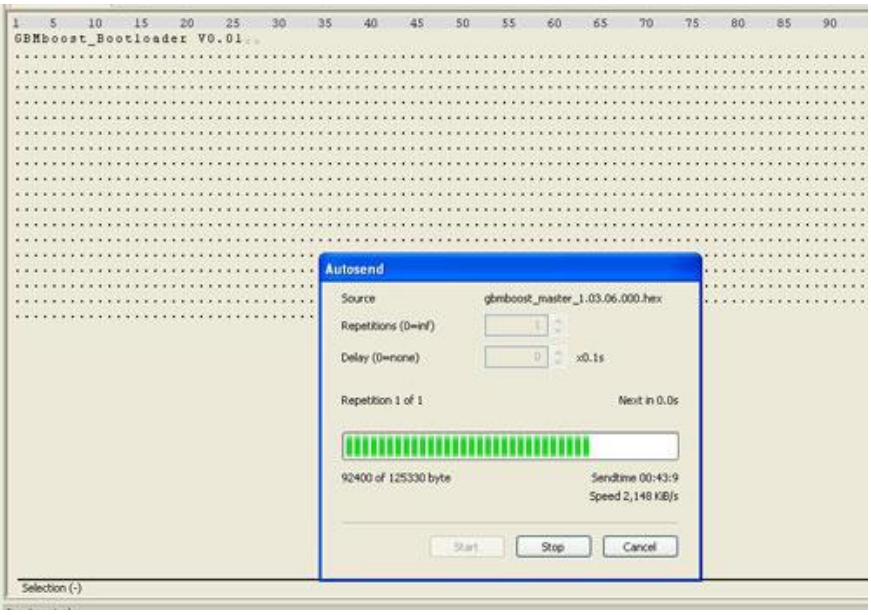
Hold down the **button** on the GBM16T and turn on the power for the replacement GBM16T.

The GBM16T answers with "GBM16T_Bootloader V?"

Step1:
Now you send a f and press Enter. The entry is made in the field of the terminal program.

The GBM16T responds with a dot.

Step2:
Now we click on the button "Send File" and select the appropriate flash firmware file (* or * 000.hex hex), and press Start.
The first part of the firmware is transmitted to the GBM16T, visible through the many points in the terminal program.



```
GBMboost_Bootloader V0.01..
```

Autosend

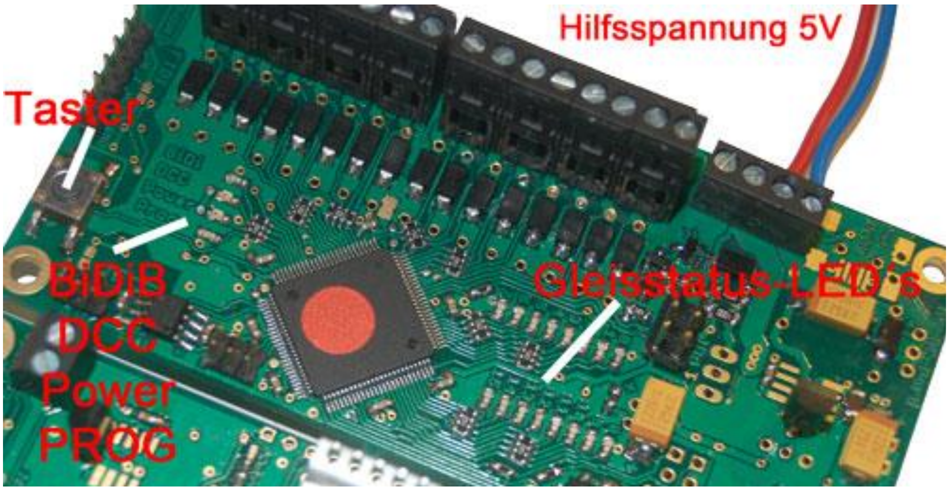
Source: gbmboost_master_1.03.06.000.hex
Repetitions (0=inf): 1
Delay (0=none): 0 x0.1s
Repetition 1 of 1 Next in 0.0s
92400 of 125330 byte Sendtime 00:43:9
Speed 2,148 KiB/s

Step3:
Now we have to transfer the eeprom. Therefore we send an **e** and confirm with **Enter**. The GBM16T responds with a dot. We choose the appropriate eeprom file (*.eeprom or 001.hex *) and press **Start**. It follows a few points back.

For GBM16T no serial number is required.

The **GBM16T** is successfully updated or placed in service, disconnect and re-connect when after the flickering auxiliary voltage / replacement power, the **power LED**.

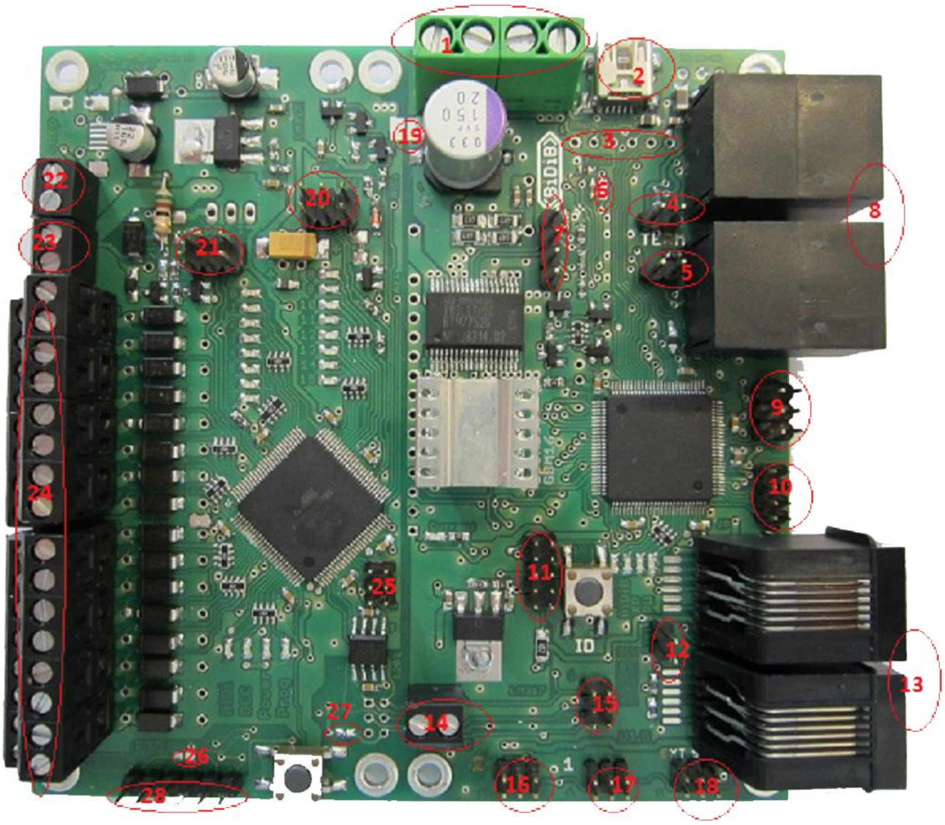
GBM16T makes the startup a self-test, which can be seen by the short duration of the **track status LEDs**.

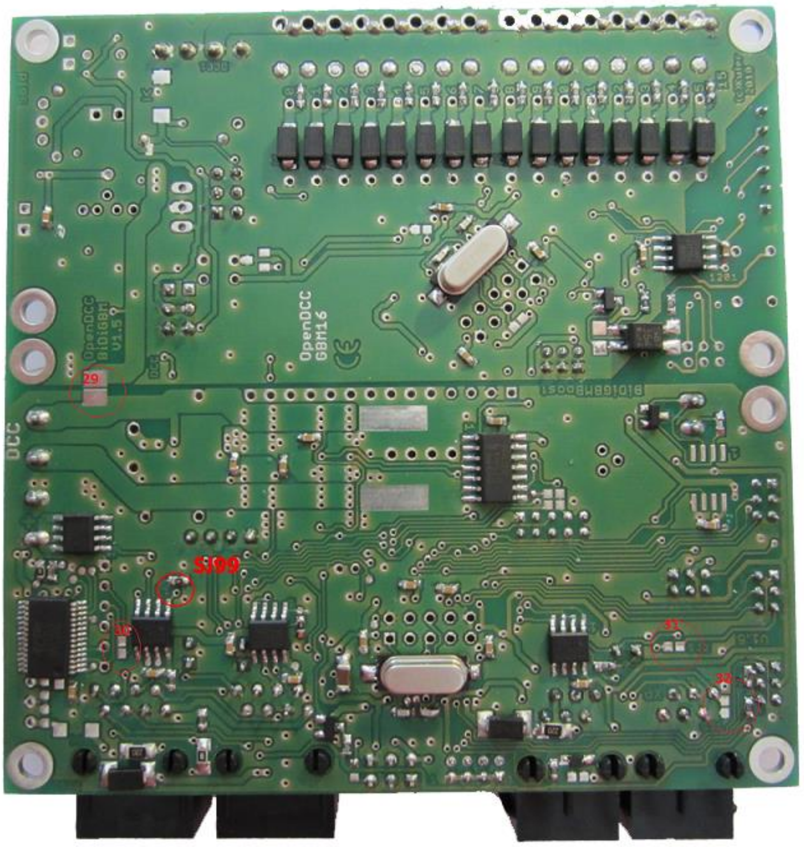


Taster = button; hilfsspannung = auxiliary voltage; Gleisstatus = track status

7 Chapter: Connectors, jumpers and solder bridges

The following are the connections, jumpers and solder bridges of OpenDCC GBM be explained to the extent necessary for commissioning and programming.





Reference	**Designation**	Description	Hint Normal mode
1	X34-1, X34-2	DC IN 14 – 20 volts 4 Amp; X34-3, X34-4 DCC to Track	
2	J90	USB UART for PC	
3	J91	Debug interface when IC90 not equipped	Not fitted
4	J5	BIDIB Bus Terminals	Open
5	J3	BIDIB Bus Terminals	Open
6	SJ10	Mode, see diagram page 4	Application
7	J8		na
8	J80, J81	BIDIB Bus Termination	Application
9	J50	PDI programming Atmega	
10	J54	modes	Application
11	J4	Optional display	
12	J72	Xpressnet Bus Terminals	Application
13	J70, J71	Xpressnet	
14	X30	DCC – in Central connection	Application
15	J51	Trackproc 1 Link	
16	J52	Trackproc 2 Link	
17	J53	Trackproc 3 Link	
18	J2	stop button	
19	SJ5	Close at supply of TP from GBMboost	
20	J10, J11	Terminal loop module (reverse loop?)	
21	JP1	PDI programming Atmega TP	
22	X2-1, X2-2	5 Volt DC in; X2-3, X2-4 DCC in	
23	X8	DCC 1 Tracks; X7 DCC Track 0 - 7	
24	X20	DCC 1 Tracks; X3 DCC Track 8 - 15	
25	J6	TP: Learn, I-Source, Bootloader	Application
26	SJ31	Debug interface	Open
27	SJ30	**GBM16T recognition**	on
28	X14	Debug interface Trackproc	
29	SJ6	Close at supply of TP from GBMboost	
30	SJ91		Open
31	SJ2		Open
32	SJ1		Open
33	J7		Application
34	**SJ99**	**Option 2 / external booster (see section 2.5)**	**Open**

Repair Service:
Assemblies to be sent for repair or inspection will be reviewed by us and repaired. In case of warranty repair for you is free. If the damage from improper assembly, installation or a result of the information in the manual differ commissioning, we are entitled to charge you for the cost of repair.

Further information is available via the e-mail address support@fichtelbahn.de.

Suggestions for improvements and information regarding errors are greatly appreciated.

The instructions and software, there is no liability for damage or a performance guarantee. I am not liable for damages that the user or others cause through the use of software or hardware or suffer. In any case I am not liable for lost sales or profits or other financial damages that may arise from the use or the use of these programs or instructions.

For inquiries, please contact our support forum!
(www.opendcc.de)

Contact:
fichtelbahn.de
Christoph Schörner
Am Dummersberg 26
D-91220 Schnaittach
support@fichtelbahn.de

Owners: © Wolfgang Kufer, Mühldorf
Author, images and graphics: Christoph Schörner Schnaittach