

MoBaLiSt

BiDiB-Knoten mit dem Schwerpunkt „Beleuchten“



Die **MoBaLiSt (ModellBahnLichtSteuerung)** von Michael Volk, ist der kleine Bruder der bekannten LightControl. Die MoBaLiSt ist ausgestattet mit einem BiDiB-Interface und kann parallel zur LightControl weitere Steuerungsaufgaben übernehmen. Die MoBaLiSt besitzt 24 Ausgänge, die frei über das BiDiB-Wizard Tool konfiguriert werden können. Die 24 Ausgänge sind aufgeteilt auf 3 Ausgangsgruppen, die je nach Bedarf von unterschiedlichen Spannungsquellen versorgt werden können.

Leistungsmerkmale:

- Eingangsspannung 5V-18V AC/DC
- 24 frei programmierbare Schaltausgänge mit je 300mA (Gesamtleistung je Chip 500mA)
- 2 universelle Eingänge
- BiDiBus-Anschluß
- Platinengröße: 50mm x 100mm

Firmware

Seit der Firmwareversion 2.0.3 hat man die Wahl zwischen drei Versionen:

STANDARD für Atmega32

- 12 Makros mit je 16 Einträgen
- 8 Accessory mit je 6 Begriffen

TURNOUT für Atmega32

- 24 Makros mit je 6 Einträgen
- 12 Accessory mit je 4 Begriffen

STANDARD für Atmega644

- 24 Makros mit je 20 Einträgen
- 12 Accessory mit je 8 Begriffen

Im Downloadfile sind alle drei Versionen enthalten.

[Firmware Download](#)

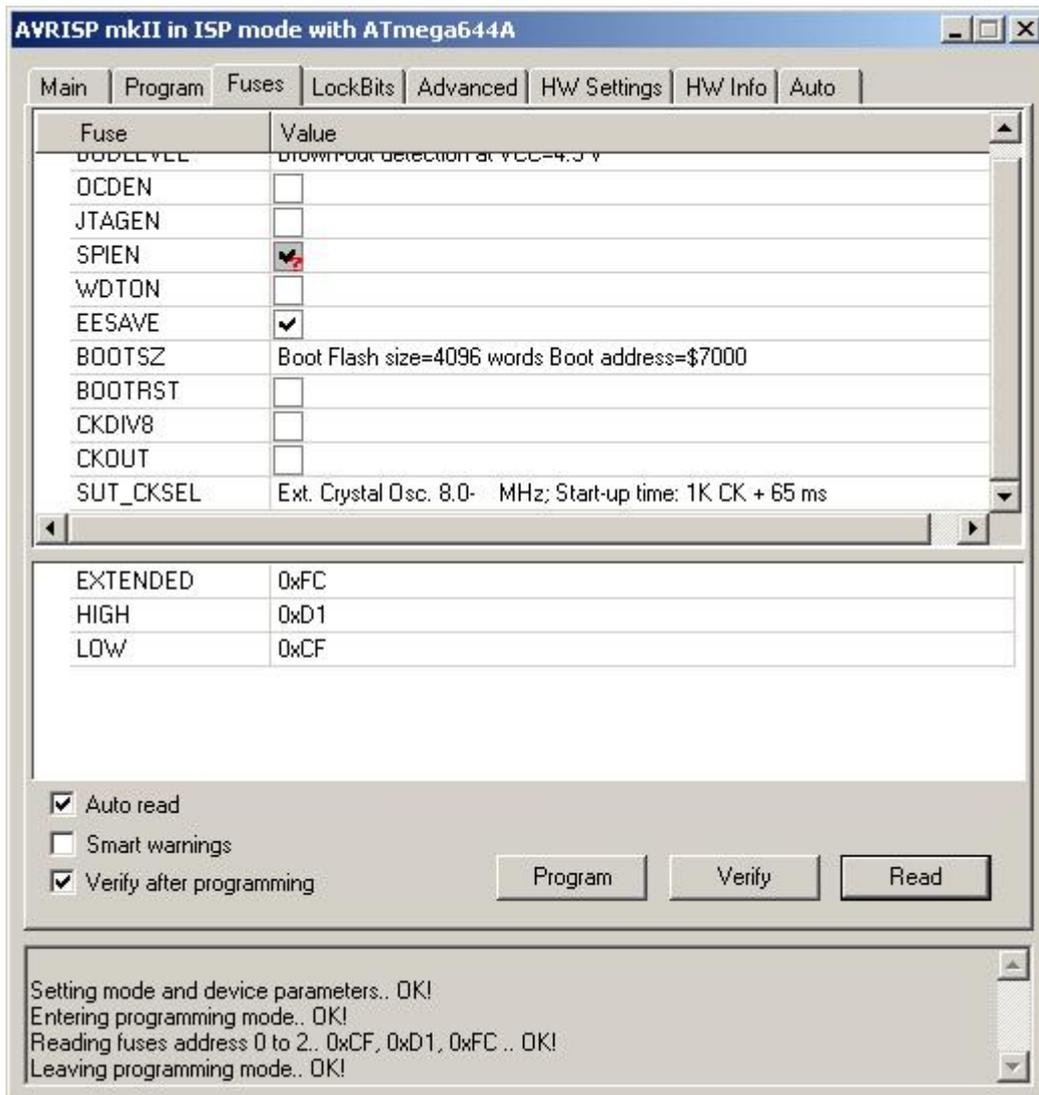
Fuses

Fuses für Atmega644

Um die MobaList64 zu flashen müssen folgende Fuses gesetzt werden (Quelle Forum):

```
BODLEVEL : Brown-out detection at VCC=4,3V
SPIEN: aktiviert
EESAVE: aktiviert
CKDIV8: deaktiviert
BOOTSZ: Boot Flash size=4096 words Boot address=$7000
SUT CKSEL: Ext.Crvstal Osc. 8.0- MHz: Start-up time: 1KCK+65ms

EXTENDED: 0xFC
HIGH: 0xD1
LOW: 0xCF
```



Programmierung mit DIAMEX ALL-AVR

Die Programmierung der MoBaLiSt mit dem DIAMEX ALL-AVR funktioniert nur mit avrdude. Nachfolgend sind die Schritte beschrieben:

libusb-win32 installieren

Damit der DIAMEX ALL-AVR von avrdude erkannt wird muss folgendes installiert werden.:

- libusb-win32: libusb-win32-bin-1.2.6.0.zip
 - Download unter <http://sourceforge.net/projects/libusb-win32/files>,
 - Installationsanleitung hier: <http://sourceforge.net/apps/trac/libusb-win32/wiki>
- avrdude-5.11:
 - Download unter <http://download.savannah.gnu.org/releases/avrdude>,
 - Manual: [http://www.nongnu.org/avrdude/user-manu ... ude_4.html](http://www.nongnu.org/avrdude/user-manu...ude_4.html)

Wenn die libusb-win32 Installation funktioniert hat gibt es im DeviceManager den Eintrag libusb-win32 devices und dort drunter den AVRISP mkII.



Falls das AVR Studio nicht installiert ist muss der Jungo-Treiber installiert werden!

Meine Schritte zum programmieren der MobaList:

```
// erase device
avrdude -p m32 -c avrisp2 -P usb -e

// flash
avrdude -p m32 -c avrisp2 -P usb -U flash:w:mobalist.hex

// eeprom
avrdude -p m32 -c avrisp2 -P usb -U eeprom:w:mobalist.eep

// serien nummer
avrdude -p m32 -c avrisp2 -P usb -U eeprom:w:MobaLiSt_V0D_XXXXXXXX.eep

// fuses
avrdude -p m32 -c avrisp2 -P usb -U hfuse:w:0xC1:m
avrdude -p m32 -c avrisp2 -P usb -U lfuse:w:0x8F:m
```

Es gibt im Diamex Forum dazu eine Anleitung:

<http://forum.diamex.de/content.php?32-ALL-AVR-ISP-Programmer>

Im Dokument erfos-isp2.pdf

(<http://forum.diamex.de/attachment.php?attachmentid=38&d=1343918313>) stehen die Schritte auch nochmal drin.

Programmierung mit ISP USBasp Programmer

Quelle: <https://forum.opendcc.de/viewtopic.php?f=29&t=5053>

Ich habe einen sehr günstigen (ca. 2 Euro) Programmer bei ebay gefunden - einfach Suche nach „usb avr isp“, dann bekommt man diese Teile angeboten. Sie haben allerdings eine 10-polige Schnittstelle, d.h. man muss noch einen Adapter auf eine 6-polige-ISP-Schnittstelle dazu bestellen oder selbst bauen ([AVR_In_System_Programmer](#)). Zeit hat man ja, die Lieferung aus China dauert ca. 3 Wochen ;)

Den zugehörigen Treiber findet man hier:

<http://zadig.akeo.ie>

→ Programm runterladen, installieren, anschließend Programmer anschließen, Programm schlägt Treiber vor → installieren

Dann muss man noch libusb-win32 installieren, um das Gerät unter avrdude ansprechen zu können:

<https://sourceforge.net/projects/libusb-win32/files/>

-> installieren, anschließend erscheint im Gerätemanager ein Unterpunkt „libusb-win32 Usb Devices“ und darunter der Programmer (USBasp) auf korrekte Jumperung der Spannung achten (meist 3,3V -

Mobalist jedoch 5V) → JP2 am Programmer entsprechend Beschriftung.

Allerdings habe ich das Teil nicht unter Atmel/AVR Studio zum laufen bekommen. Dafür funktioniert es mit dem kommandozeilenbasierten Tool „avrdude“, was man hier herunterladen kann:

<http://download.savannah.gnu.org/releases/avrdude/>

Letzte Version runterladen und in ein Verzeichnis deiner Wahl entpacken.

Die Firmware-Dateien findet man [hier](#).

*.000.hex: Flash-Daten *.001.hex EEPROM-Daten

Die zugehörige Seriennummer kann man unter [BiDiB Seriennummern-Generator](#) generieren. (Zugang mit Zugangsdaten des OpenDCC-Forums)

Anschließend die 3 Flashdateien (Flash, EEprom und SN) in das avrdude-Verzeichnis kopieren. Die Endung (*.hex oder *.eep) spielt anscheinend keine Rolle.

DOS-Fenster öffnen (cmd) und zum Avrdude-Verzeichnis wechseln [cd \xxx\avrdude\]

Nun den Programmer anschließen (auf Pin 1 achten) und den Mobalist mit 12VDC versorgen (grüne LED27 leuchtet).

Nun erfolgt die Übertragung von Firmware, SNr und das Setzen der Fuses wie folgt:

```
// erase device
avrdude -p m32 -c USBasp -P usb -e

// flash
avrdude -p m32 -c USBasp -P usb -U flash:w:mobalist_STANDARD_2.00.07.000.hex

// eeprom
avrdude -p m32 -c USBasp -P usb -U
eeprom:w:mobalist_STANDARD_2.00.07.001.hex

// seriennummer
avrdude -p m32 -c USBasp -P usb -U eeprom:w:MobaLiSt_V0D_xxxxxxx.001.hex

// fuses
avrdude -c usbasp -p m32 -P usb -U hfuse:w:0xC1:m
avrdude -c usbasp -p m32 -P usb -U lfuse:w:0x8F:m
```

Wenn alles ok ist, sollte nun die rote LED D26 flackern.

Der Programmer kann nun abgezogen und die Verbindung zum Bidib-Bus angeschlossen werden. Vorher noch den Terminierungsjumper (JP7) setzen und ggf. den an dem letzten Bidib-Gerät, wo ihr euch angeschlossen habt, entfernen.

Wenn alles ok ist, sollte nun die grüne LED29 an den Bidib-Buchsen leuchten und nach Verbindungsaufnahme die orange LED D25 neben der Stromversorgung (P8). Im Bidib-Wizard sollte nun die Mobalist angezeigt werden. Nun kann man die Funktion der LED-Ausgänge testen: Doppelklick auf das Mobalist-Symbol - Reiter Lichtausgänge - ganz rechts: Testen (ein/ausschalten).

Programmierung mit Ponyprog und Selbstbauadapter

Diese Anleitung bezieht sich auf den seriellen Ponyprog Adapter aus dem [Opendcc Versand von Hanno Bolte](#).

Der Adapter kann natürlich auch selbst gebaut werden, notwendig sind: 3x Widerstand 1kOhm, 3x Z-Diode 4,7V, 1x Kondensator 1nF, 1x DB9 Buchse, ein kleines Stück 6-poliges Flachbandkabel und die dazu passende Pfostenbuchse 2x3 mit Rastermaß 2,54mm.

Achtung: Dieser Adapter ist nur für Prozessoren geeignet, die mit 5V betrieben werden, die meisten Baugruppen aus dem BidiB Projekt arbeiten mit einer Prozessorspannung von 3,3V. Beim Versuch diese Prozessoren damit zu programmieren könnten sie beschädigt oder zerstört werden!

Nach meinem Wissen sind aus dem BidiB Projekt nur die Baugruppen Mobalist, S88 BidiB Bridge und DCC-Pola mit 5V Prozessoren ausgestattet.

Für die Programmierung mit Ponyprog wird natürlich auch das Programm selbst benötigt.

Windows User mit 9x/ME/NT/2000/XP können dieses direkt von der [Seite des Programmierers](#) herunterladen.

Für Windows 7 (andere neue Betriebssysteme müßte jemand testen) benötigt man eine [inoffizielle, gepatchte 64 Bit Version mit einem eigenen Treiber für den Zugriff auf die serielle Schnittstelle](#).

Nach der erfolgreichen Installation muß man erst ein paar Settings vornehmen. Menü Setup ⇒ Interface Setup. Hier die korrekte serielle Schnittstelle einstellen, SI-Prog API auswählen und den Haken bei Invert Reset setzen. Mit ok abspeichern. Auch sollte man kurz unter Setup ⇒ Calibration eine Kalibration der Schnittstelle durchführen. Das muß nur beim 1. mal gemacht werden.

Nun wird es interessant. Der korrekte Prozessor wird im Menü Device ⇒ AVR micro ⇒ ATmega32 ausgewählt. Nun den Mobalist mit dem Adapter verbinden und mit Strom versorgen. Dabei sollte die Power LED leuchten. Außer der Stromversorgung und dem seriellen Kabel sollten keine weiteren Verbindungen bestehen.

Die Config Bits werden gesetzt: Command ⇒ Security and Configuration Bits, dort die Häkchen wie gezeigt setzen.



Nach doppelter Kontrolle mit Write auf den Prozessor schreiben.

Vor dem Programmieren wird der Prozessor mit CTRL-E bzw. im Menü mit Command ⇒ Erase komplett gelöscht.

Nun folgende Dateien bereithalten:

- 1.) Firmware z.B. mobalist_STANDARD_2.00.03.000.hex
- 2.) EEPROM Daten z.B. mobalist_STANDARD_2.00.03.001.hex
- 3.) Seriennummer z.B. MobaLiSt_V0D_P6C00xyza.001.eep (xyza ist die eigentliche Seriennummer).

Das Firmwarefile wird als erstes eingespielt. Daten im Menü File ⇒ Open Program (FLASH) File auswählen, dann mit Command (Write Program (FLASH) schreiben. Die Sicherheitsabfrage mit Yes bestätigen. Dann etwas warten, am Ende wird die geschriebene Datei noch mit der am Prozessor verglichen. Als nächstes folgt das EEPROM File, da Ponyprog aber nur den ganzen EEPROM auf einmal schreiben kann muß man dort erst noch die korrekte Seriennummer eintragen. Würde man die Files nacheinander (Firmware, EEPROM, Seriennummer) schreiben, so überschreibt die Seriennummer das komplette EEPROM wieder. Ergebnis ist ein Mobalist, der sich zwar korrekt am BidiB Bus meldet, sich sonst aber eigenartig verhält (alle Feature Register stehen auf 255).

Das EEPROM File im Menü ⇒ Open Data (EEPROM) File auswählen. Evtl. muß rechts unten die Dateierweiterung *.eep ausgewählt werden. Nun stellt man die Seriennummer-Config im Menü Utility ⇒ SerialNumber Config wie folgt ein:



Address: 0x83FE

Size: 2

Value: Seriennummer dezimal

Haken bei Auto Increment herausnehmen

Die dezimale Seriennummer kann man errechnen, indem man das File MobaLiSt_V0D_P6C00xyza.001.eep nimmt und im Windows Taschenrechner den Programmierer Modus auswählt. Dann links auf Hex klicken und die 4 Stellen (xyza) aus dem Dateinamen eingeben. Habe das hier exemplarisch mit der Seriennummer AAAA gemacht. Dann auf Dez klicken und schon erhält man die dezimale Seriennummer, in diesem Fall 43690. Die Nummer bei Value eingeben.

Um die Nummer nun im EEPROM Bereich zu speichern im Menü auf Utility ⇒ Set SerialNumber oder mit CTRL-N.

Jetzt kann das EEPROM korrekt beschrieben werden: Menü Command ⇒ Write Data (EEPROM).

Zum Schutz des EEPROM vor Überschreiben bei einem Prozessor-Update sollte man jetzt nochmals in den Config Bits eine Änderung vornehmen: Menü Command ⇒ Security and Configuration Bits, dort den Haken bei EESAVE setzen.

FERTIG. Nach dem Abstecken des Programmierkabels sollte die rote LED am Mobalist flackern, wenn das Kabel zum BidiB angesteckt wird müssen auch die grüne Bus-LED und die orange LED leuchten.

From:

<https://forum.opendcc.de/wiki/> - **BiDiB Wiki**

Permanent link:

<https://forum.opendcc.de/wiki/doku.php?id=mobalist>

Last update: **2018/01/08 09:29**

