

# LED-IO-24

## BiDiB-Knoten mit dem Schwerpunkt „Beleuchten“, „Input“ und „Belegtmeldung“



Die **LED\_IO\_24** ist eine Weiterentwicklung der MoBaList. Die LED\_IO\_24 ist ausgestattet mit einem BiDiB-Interface und besitzt 24 Ausgänge/Eingänge, die frei über das BiDiB-Wizard Tool konfiguriert werden können. Die 24 Ausgänge/Eingänge erlauben den Anschluss von LEDs für Beleuchtungsaufgaben. Als Eingang konfiguriert können sie als Eingänge verwendet werden oder als Belegtmelder konfiguriert werden.

## Leistungsmerkmale:

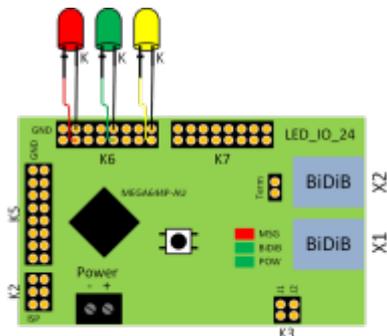
- Eingangsspannung 8V-20V DC
- 24 frei programmierbare Ausgänge/Eingänge
- Jeder Port als Ausgang konfigurierbar für je eine LED mit maximal 20mA
- Jeder Port als Eingang oder Belegtmelder konfigurierbar
- BiDiBus-Anschluß
- Platinengröße: 50mm x 80mm

## LED Anschluss:

Die 24 LED Ausgänge verfügen auf dem Modul über je einen internen 100R Vorwiderstand. Die LEDs können somit direkt an die Baugruppenausgangspins der Stiftleisten K5, K6 und K7 angeschlossen werden. Dabei kommt die Kathode der LED jeweils an den zum Platinrand zeigenden Pin von K5 - 7 und die Anode zum innen liegenden Anschlusspin.

Die LED Betriebsspannung auf der Baugruppe beträgt 4,5V. Damit lassen sich sämtliche gängigen LEDs ohne zusätzliche externe Widerstände direkt an der Baugruppe anschließen. Die internen 100R Vorwiderstände je Ausgang, schützen diesen auch bei einem direkten Kurzschluss der Ausgangspins vor Zerstörung.

**ACHTUNG**, an dieser Baugruppe kann je Ausgang genau eine LED angeschlossen werden. Dabei wird die LED direkt an die beiden Ausgangspins angeschlossen. Sie darf NICHT an eine weitere externe Betriebsspannung angeschlossen werden!



Alternativ lassen sich die 24 LED Ausgänge einzeln umschalten damit ein Anschluss gegen die Anode erfolgt. In diesem Fall wird die 5V Spannung des Bausteins an einer geeigneten Stelle abgegriffen (bspw. Pin 2 des ISP) und mit der Anode der LED verbunden. Die Kathode wird dann mit dem inneren Anschluss des jeweiligen Ports verbunden. Über CV 180 - 182 wird dann definiert dass die LED gegen die 5V geschaltet wird.

---

## Eingang:

Folgt

---

## Belegtmelder:

Folgt

---

## Portumschaltung:

Jeder einzelne der 24 Ports kann von LED Ausgang auf Eingang umgeschaltet werden. Die Umschaltung wie folgt durchführen:

### LED Ausgang -> Eingang

Wenn der Port aktuell als LED Ausgang aktiv ist im BiDiB Wizard auf den Reiter Eingang wechseln und beim gewünschten Port rechts klicken. Im Popup Menü auf „Port umschalten“ klicken und die Umschaltung bestätigen.

Ein auf Eingang umgestellter Port kann zusätzlich über die CV 171 - 173 von Eingang auf Belegtmelder umgestellt werden.

## Eingang -> LED Ausgang

Wenn der Port aktuell als Eingang aktiv ist im BiDiB Wizard auf den Reiter Licht wechseln und beim gewünschten Port rechts klicken. Im Popup Menü auf „Port umschalten“ klicken und die Umschaltung bestätigen.

---

## Firmware

Die Beschreibung zur Einstellung der Fuses des Prozessors und die Installation der Firmware ist im Unterabschnitt zu finden. Dies wird normalerweise nicht benötigt, da Updates über den BiDiB-Wizard eingespielt werden können.

### [Firmware Beschreibung](#)

Die Firmware (bestehend aus Bootloader und eigentlicher Firmware) ist im Unterabschnitt zu finden.

### [Firmware Download](#)

---

## 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:led_io_24.hex

// eeprom
avrdude -p m32 -c avrisp2 -P usb -U eeprom:w:led_io_24.eep

// serien nummer
avrdude -p m32 -c avrisp2 -P usb -U eeprom:w:LED_IO_24_V0D_XXXXXXXX.eep
```

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 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, LED\_IO\_24, 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.

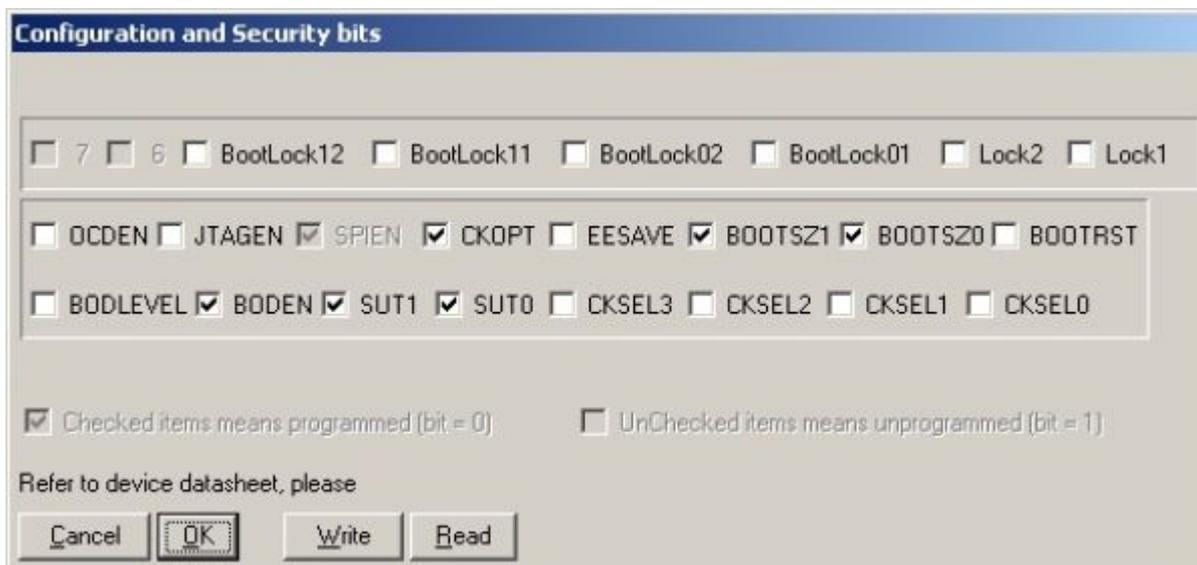
Für Windows 7 (andere neue Betriebssysteme müßte jemand testen) benötigt man eine [inoffizielle, gepatchte 64 Bit Version mit einem eigenen Treiber für den Zugriff auf die serielle Schnittstelle](#).

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\_MOBALED\_2.01.10.000.hex
- 2.) EEPROM Daten z.B. mobalist\_MOBALED\_2.01.10.001.hex
- 3.) Seriennummer z.B. LED\_IO\_24\_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.

---

## Selbstbauanleitung

Michael Engelen hat eine Aufbau Anleitung für den Selbstbau dieser Baugruppe erstellt. Diese ist hier zu finden:

[LED\\_IO\\_24\\_Aufbauanleitung](#)

From:

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

Permanent link:

[https://forum.opendcc.de/wiki/doku.php?id=led\\_io\\_24&rev=1486678272](https://forum.opendcc.de/wiki/doku.php?id=led_io_24&rev=1486678272)

Last update: **2017/02/09 23:11**

