PIO48-OCA

20. September 2011

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Hinweis

Die Angaben in diesem Datenblatt sind von uns auf Übereinstimmung mit der beschriebenen Hardware bzw. Software gerpüft worden.
Aus verschiedenen Gründen sind Abweichungen dennoch nicht ausgeschlossen, weswegen wir für eine vollständige Übereinstimmung keine Gewährleistung übernehmen können.
Alle zukünftigen, relevanten Änderungen werden im Abschnitt 6 dokumentiert.
Für Anregungen oder Verbesserungsvorschläge sind wir jederzeit aufgeschlossen.

1 Eigenschaften

Das NetPort PIO48OCA ist eine universelle Steuerung, die durch den Endanwender auf viele Bedürfnisse angepasst werden kann. Der Endanwender wird in die Lage versetzt modernste Webtechnologie mit der Steuerung von externer Hardware zu verbinden.
Ein integrierter Skriptinterpreter ermöglicht die Implementierung komplexer, autarker Steuerungs- und Regelungsaufgaben. Ein Webserver bietet die Möglichkeit eigene Webseiten anzuzeigen, die in Interaktion mit der Hardware stehen können. Vielfältige Konfigurationsmöglichkeiten lassen die Anwendung im Industriellen- und Konsumerbereich zu. Auf der Hardwareebene kann das Modul einfach um zusätzliche Komponenten erweitert werden.

Mögliche Anwendungen sind:

- Steuerungen aller Art.
- Fernsteuerung von Aktoren über ein Webinterface.
- Steuerung von Modellbauservos per Ethernet.
- Messung von physikalischen Grössen mit Webinterface.
- Inbetriebnahme- und Test bei der Entwicklung elektronischer Komponenten.

Das PIO48OCA verfügt über **vier** Open Kollektorausgänge, die für einen maximalen Strom I_{OX} (Siehe Tabelle 3) ausgelegt sind. Weiterhin sind **8** digitale I/O Anschlüsse vorhanden, denen per Software eine Funktion zugeordnet werden kann.

Weitere Merkmale sind:

- Piezo Signalgeber
  Frequenz im Bereich $fPiezo_{Min}$ - $fPiezo_{Max}$ per Software einstellbar. Lautstärke per Software einstellbar. (0..100%).
- Ethernet Schnittstelle
- USB Schnittstelle
- DSUB15 Verbinder für Ein- und Ausgänge
- **C Interpreter**
  (Bis auf wenige Ausnahmen ISO C Kompatibel. Zusätzlich werden einige C++ Konstrukte unterstützt.)

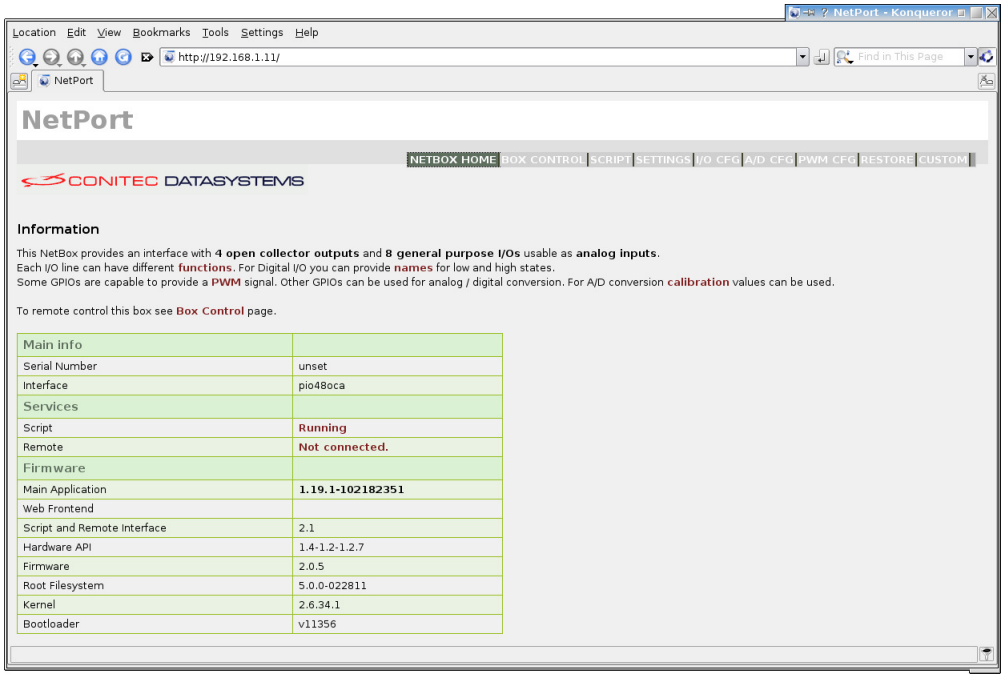
2 Bedienung

Das PIO Modul kann auf verschiedene Arten verwendet werden. Siehe Sofwareschnittstellen (3.1).

2.1 Die Weboberfläche

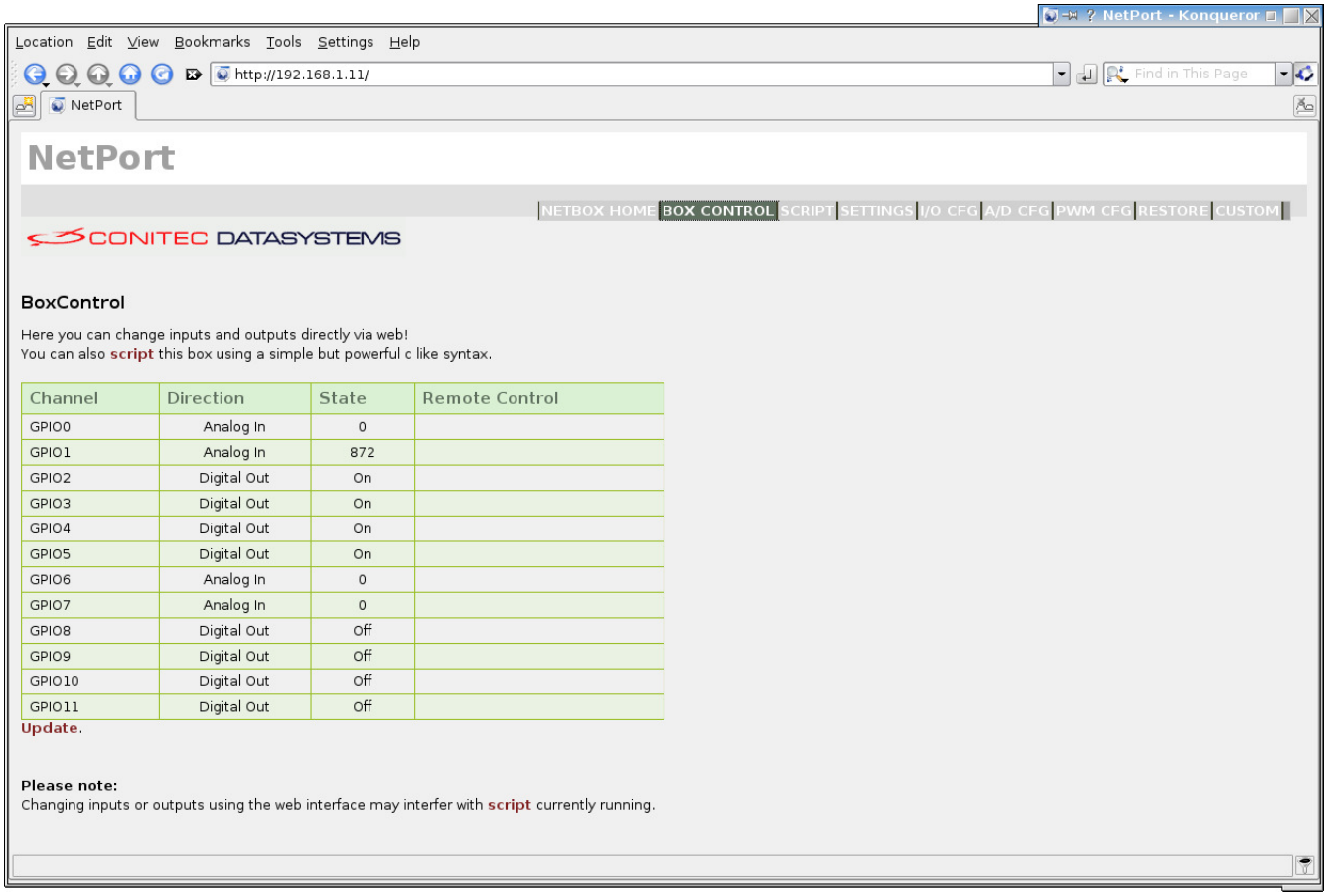
Sie erreichen das Modul, indem Sie dessen IP-Adresse in Ihren Browser eingeben. Zum Beispiel (Werkseinstellung):

http://192.168.1.11



Kurz darauf erscheint die Webseite.

2.1.1 BOX CONTROL



Auf dieser Seite können Sie die Ausgänge kontrollieren und die Eingänge einlesen.

Hinweis

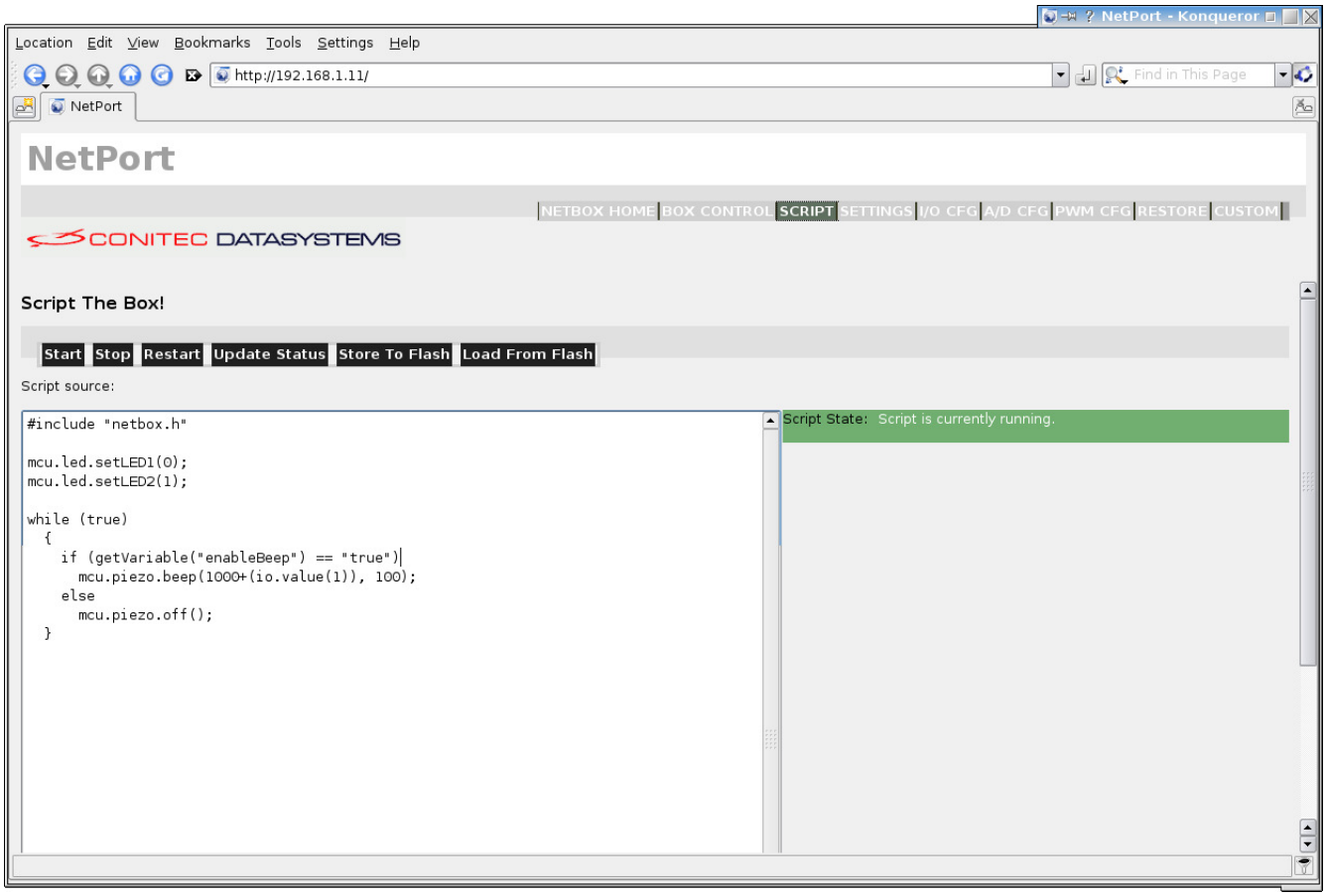
Sie können die Webseite aktualisieren, indem Sie "Update" betätigen.

Je nach Einstellung unter Settings (Siehe Einstellungen 2.1.3) werden die Signalrichtungen entsprechend dargestellt.

Bitte beachten Sie, dass die manuelle Steuerung der Box auch dann erlaubt ist, wenn bereits ein Skript auf der Box ausgeführt- oder die Box anderweitig ferngesteuert wird.

2.1.2 SCRIPT

In diesem Bereich der Webseite können Sie einfache C-Skripte auf die Box laden und ausführen. Die Skripte können dazu dienen, einfache Abläufe zu realisieren oder auch komplexe Steuerungsaufgaben zu erledigen. Skripte können direkt per Weboberfläche eingegeben und gespeichert werden, so dass sie beim nächsten Start automatisch ausgeführt werden. (Siehe Einstellungen 2.1.3).



Syntax- oder Laufzeitfehler werden ebenfalls in der Weboberfläche dargestellt.

2.1.3 SETTINGS (Einstellungen)

Hier können die Einstellungen der Box verändert werden. Beachten Sie, dass neue Einstellungen erst wirksam werden, wenn “Apply Settings” betätigt wird.

2.1.3.1 Autostart

Diese Funktion bezieht sich auf das Skript. Wenn sie Aktiv ist wird das Skript sofort nach dem Einschalten ausgeführt.

2.1.3.2 When script fails : Restart

Wenn diese Funktion eingeschaltet ist, wird das Skript nach auftreten eines Fehlers sofort neu gestartet. (Fehler können Laufzeitfehler oder auch Syntaxfehler sein.)

Hinweis

Diese Option ist bei Auslieferung eingeschaltet. Für die Entwicklung und zum Test von Skripten sollte diese Funktion ausgeschaltet sein.

2.1.3.3 When script fails: Warning Beep

Wenn diese Funktion eingeschaltet ist, ertönt ein aktustisches Signal, sobald ein Skriptfehler aufgetreten ist.

2.1.3.4 When script terminates: Restart script automatically.

Wird das Skript regulär beendet und diese Option ist eingeschaltet wird es automatisch neu gestartet.

2.1.3.5 I/O Names

Hier können Namen für die GPIOs eingetragen werden. Weiterhin ein Symbol zur Kennzeichnung für einen Aktiven Ausgang (“Name for on”), sowie eine Bezeichnung für einen passiven Ausgang (“Name for off”). Das Feld “Negated” markiert einen invertierten Ausgang.

2.1.3.6 Initial Direction

Hier kann für jedes Pin getrennt eine Funktion festgelegt werden. Es handelt sich um die Funktion, die automatisch nach dem Bootvorgang bzw. beim Start des Skriptes ausgewählt wird. Die Pinfunktion kann später auch durch das Skript oder per Fernsteuerung verändert werden.

Siehe auch 3.4.2.

2.1.3.7 IO/CFG

Hier können Namen für Pinzustände vergeben werden. Die Namen werden in BoxControl verwendet um den Pinzustand darzustellen. Weiterhin kann mit der “Initial State” Checkbox für jedes I/O Pin voreingestellt werden, ob das Pin aktiv (high) oder passiv (low) sein soll.

Hinweis

Diese Einstellungen werden nur für I/O Pins angewendet, die als digitaler Ausgang definiert sind.

2.1.3.8 A/D CFG

Unter A/D CFG können Kalibrierwerte eingegeben werden, die im Skript ausgewertet werden können. (Dieses Merkmal wird ab Version **2.2** der “Script und Remote Interface” Software verfügbar sein.

2.1.3.9 PWM CFG

Hier können die Software PWMs konfiguriert werden. Einstellbar sind:

- Periodendauer
- On Zeit

Beide Werte in in µs einzugeben.

2.1.4 Sichern und Wiederherstellen

Ab Firmware Version 1.14 wird die Funktion “Restore” zur Verfügung gestellt. Diese Funktion ermöglicht es dem Benutzer alle Einstellungen der Box als Datei zu sichern und später zurückzuschreiben.

2.1.4.1 Einstellungen speichern

Um die Einstellungen zu sichern genügt ein Klick auf “SETTINGS”▷”Download Configuration”. Je nach Browser wird die Datei sofort heruntergeladen oder es kommt eine Aufforderung zum Speichern der Datei. Die Datei mit der Endung “.netbox” enthält alle Einstellungen wie IP-Adresse, I/O Konfiguration und auch das Skript. Zusätzlich wird die Seriennummer des PIO Moduls gespeichert.

2.1.4.2 Gespeicherte Einstellungen zurückschreiben

Das Zurücksichern der Einstellungen geschieht über die Weboberfläche im Im Menupunkt “RESTORE”. Hier kann angebeben werden, welche Inhalte wiederhergestellt werden sollen:

- “Restore Script”
  Das aktuelle Skript wird das in der Konfigurationsdatei (.netbox) gespeicherte Skript ersetzt.
- “Restore Settings”
  IP Adresse, I/O Konfiguration und alles, was sich unter “SETTINGS” befindet wird durch die Konfigurationsdatei ersetzt.
- “Restore Custom Website”
  Die benutzerdefinierte Webseite wird zurückgesichert.
- “Restore only when serial number matches”
  Wenn diese Option angeklickt ist wird zunächst geprüft, ob die Seriennummer des aktuellen Gerätes mit der Seriennummer der Konfigurationsdatei übereinstimmen. Nur wenn das der Fall ist werden Änderungen vorgenommen. Sollten die Seriennummern **nicht** übereinstimmen wird eine Fehlermeldung ausgegeben und **nichts** geändert.
- “Reboot after restoring data”
  Wenn diese Option aktiv ist, wird die Box nach Anwendung der Konfiguration automatisch neu gestartet.

Sobald Sie die Datei auf der Webseite angegeben haben und “Restore” drücken werden die Einstellungen übernommen und die Box ggf. neu gestartet.

2.2 Benutzerspezifische Webseite

Das Modul erlaubt es dem Anwender eine eigene Webseite zu definieren und auf dem Modul abzulegen. Diese Webseite (“Custom Pages”) erscheint je nach Einstellung sofort auf der IP Adresse der Box oder im Bereich “CUSTOM” auf der Hauptseite.

2.2.1 Möglichkeiten

Die benutzerspezifische Webseite erlaubt es mit Hilfe des Netports eine vollständige, webgesteuerte Steuerung zu implementieren. Die Webseite kann dabei aus mehreren Dateien bestehen. (Bilder, StyleSheets, JavaSkript, ...)

Mit Hilfe definierter Schnittstellen kann über JavaSkript auf die I/O Ports und die Einstellungen zugegriffen werden. Weiterhin ist ein Datenaustausch zwischen JavaSkript und dem Box-Skript möglich. Weiteres dazu im Dokument “netPortWeb.pdf”.

2.2.2 Grenzen

Das Netportmodul stellt für die Webseite einen Speicherplatz von 256kBzur Verfügung.

2.2.3 Entwickeln der Benutzerspezifischen Webseite

Die Benutzerspezifische Webseite wird beim Systemstart in das FTP Verzeichnis unter “custom-html” geladen. Sofern sich dort eine Datei mit dem Namen “index.html” befindet und die Einstellung “Show Custom Pages as default” gesetzt ist, wird diese Datei Hauptseite verwendet.

Sie können Ihre eigene Webseite mit Hilfe eines FTP Clients auf das Netport Modul laden.

2.2.4 Speichern der Webseite

Im “SETTINGS” Bereich befindet sich ein Schalter “Store Custom Website”. Drücken Sie diesen Schalter, um alle Dateien im “custom-html” Vezeichnis dauerhaft zu speichern.

Hinweis

Die aktuell gespeicherte Webseite wird dadurch unwiederbringlich ersetzt. Machen Sie sich ggf. Sicherheitskopien des "custom-html" Verzeichnisses.

2.3 Steuerung per Skript

In diesem Kapitel finden Sie einige Skriptbeispiele. Sie können Diese Skriptbeispiele heraus kopieren und direkt in der Weboberfläche Testen.

2.3.1 Steuerung der Ausgänge

2.3.1.1 Blinker

Die folgenden Beispiele setzen voraus, dass GPIO1 als Ausgang konfiguriert ist.

```
#include <netbox.h>

while(true)
  {
    io.set(1, true);
    pause(100);
    io.set(1, false);
    pause(200);
  }
```

Quelle: ./snippets/blink.msl

Abbildung 1: Ein einfacher Blinker Version1

Dieses Beispiel lässt den Ausgang GPIO1 (DSUB15 Buchse Pin 9) blinken. (100ms an, 200ms aus).

2.3.1.2 Lauflicht

Die folgenden Beispiele setzen voraus, dass GPIO1-GPIO6 als Ausgänge konfiguriert sind.

```
#include "netbox.h"

void setMask(int iMask)
   {
     io.set(1, bool(iMask & 1));
     io.set(2, bool(iMask & 2));
     io.set(3, bool(iMask & 4));
     io.set(4, bool(iMask & 8));
     io.set(5, bool(iMask & 16));
     io.set(6, bool(iMask & 32));
   }



int iDir = 1;
int iIdx = 0;
while (true)
  {
    iIdx += iDir;
    if ((iIdx >= 5) ||(iIdx == 0))
      iDir *= -1;
    setMask(1 << iIdx);
    pause(100);
  }
```

Quelle: ./snippets/lauflicht2.msl

Abbildung 2: Noch ein Lauflicht.

Es werden nacheinander alle LEDs eingschaltet. (Wandernder Punkt.)

2.3.2 Einlesen eines digitalen Eingangs

```
#include "netbox.h"

while (true)
  {
    bool boState = io.get(2);
    io.set(1, boState);
  }
```

Quelle: ./snippets/input1.msl

Abbildung 3: Einlesen eines Eingangs.

Dieses Beispiel setzt O1 in Anhängigkeit des ersten Eingangs. (Kopiert das Signal...)

2.3.3 Einlesen eines analogen Eingangs

```
#include "netbox.h"

while (true)
  {
    bool boState = io.value(2) > 800;
    io.set(1, boState);
  }
```

Quelle: ./snippets/input2.msl

Abbildung 4: Einlesen eines Eingangs.

Sobald der analoge Wert von IO2 den skalaren Wert 800 überschreitet wird der Ausgang IO1 eingeschaltet.

2.4 Konfiguration der I/O Signale per Skript

Die Konfiguration der I/O Ports kann auch per Skript erfolgen. Hierzu sind einige Konstanten definiert.

```
// gpioCaps.h
enum GPIOConfig
  {
    gpioAnalogInput3v3                      = 32,
    gpioAnalogInput1v1                      = 33,
    gpioDigitalInputLogic                   = 16,
    gpioInvertedDigitalInputLogic           = 272,
    gpioDigitalOutputLogic                  = 64,
    gpioInvertedDigitalOutputLogic          = 320,
    gpioInvertedDigitalOutputLogicSoftPWM   = 832,
    gpioDigitalOutputLogicSoftPWM           = 576,
    gpioDigitalOutputOC                     = 65,
    gpioInvertedDigitalOutputOC             = 321,
    gpioDigitalOutputOCSoftPWM              = 577,
    gpioInvertedDigitalOutputSoftPWM        = 833
  };
```

Quelle: ./snippets/gpioCaps.h

Abbildung 5: Konstanten zur I/O Konfiguration.

Hinweis

Bitte beachten Sie, dass nicht jeder GPIO alle Möglichkeiten der Konfiguration zulässt.

Beispiel:

Das folgende Beispiel konfiguriert GPIO0 und GPIO7 als analogen Eingang. GPIO2 wird als digitaler Ausgang konfiguriert. GPIO1, GPIO3 - GPIO6 werden als digitale Eingänge konfiguriert.

```
#include <netbox.h>

io.setFunction(0, gpioAnalogInput3v3);
io.setFunction(1, gpioDigitalInputLogic);
io.setFunction(2, gpioDigitalOutputLogic);
io.setFunction(3, gpioDigitalInputLogic);
io.setFunction(4, gpioDigitalInputLogic);
io.setFunction(5, gpioDigitalInputLogic);
io.setFunction(6, gpioDigitalInputLogic);
io.setFunction(7, gpioAnalogInput3v3);
```

Quelle: ./snippets/ioCfg1.msl

Abbildung 6: Dynamische Konfiguration von I/Os.

Zu beachten ist, dass die Portrichtungen sofort gültig sind. Sie werden jedoch nicht (wie bei den Einstellungen) abgespeichert. Die skriptgesteuerte permanente Änderung der Grundeinstellungen wird in einem separaten Dokument abgehandelt.

2.5 Signalgeber

2.5.1 Ein einfaches Signal

Um einen Ton zu erzeugen:

```
#include "netbox.h"

mcu.piezo.beep(1000, 100);
pause(200);
mcu.piezo.off();
// Wait forever...
while (true) pause();
```

Quelle: ./snippets/piezo1.msl

Abbildung 7: Ein kurzer Piep mit 1KHz mit voller Lautstärke, 100ms lang.

2.5.2 Ein abklingendes “Ding”

```
#include "netbox.h"

for (int i = 0; i < 100; ++i)
  {
    mcu.piezo.beep(1000, 100-i);
    pause(1);
  }
mcu.piezo.off();
// Wait forever...
while (true) pause();
```

Quelle: ./snippets/piezo2.msl

Abbildung 8: Ein abklingendes, akutisches Signal.

2.6 Fernsteuerung über Remote Skripte

Ab Firmware Version 1.14 erlaubt die NetBox die Ausführung von Skripten, die per Netzwerkverbindung übertragen wurden. Dieses Merkmal ist zur einfachen Fernsteuerung der Box vorgesehen.

Die Box bedient hierzu den TCP Port (1233). Ein- oder mehrere Clients können sich zu diesem Port verbinden, ein Skript hinsenden und somit die Box fernsteuern. Dabei können prinzipiell die gleichen Skripte verwendet werden, wie sie auch im vorherigen Kapitel gezeigt wurden - mit zwei Ausnahmen:

1. #include <netbox.h> ist nicht notwendig und kann weggelassen werden.
2. Die Endlosschleife im unteren Teil des Programms kann / sollte weggelassen werden, damit das Kommando beendet werden kann.

Alle Skripte (Remote Skripte und das Box Skript) werden quasiparallel ausgeführt. Wenn sich ale mehrere Clients verbinden können also auch mehrere Kommandos (z.B. Berechnungen) parallel zueinander ausgeführt werden. Jedes Skript hat vollen Zugriff auf alle Resourcen wie I/O Leitungen.

Hinweis

Die Fernsteuerung über Remote Skripte ist nur aktiv, wenn unter Settings "Allow remote script" aktiv ist.

2.6.1 Ablauf einer Fernsteuerungssitzung

Es gibt zwei Möglichkeiten einer Fernsteuerungssitzung:

1. **Kommandoorientiert**
   Hierbei wird nur ein Kommando (oder eine kurze Sequenz) an die Box gesendet. Die Verbindung wird durch die Box geschlossen.
2. **Interaktiv**
   Es können nacheinander mehrere Kommandos (oder Sequenzen) gesendet werden. Die Verbindung wird durch den Client beendet.

Die folgenden Abbildungen verdeutlichen das Vorgehen. (Es wurde exemplarisch die IP Adresse 192.168.1.12 gewählt.)

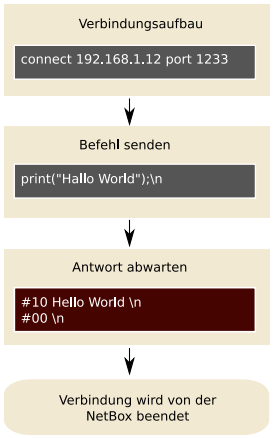


Abbildung 9: Kommandoorientierte Fernsteuerung

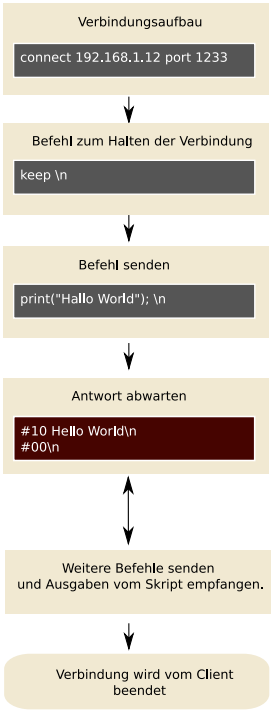


Abbildung 10: Interaktive Fernsteuerung

2.6.2 Beispiele für Fernsteuerskripte

In diesem Abschnitt finden Sie Beispiele für einfache Fernsteuerskripte. Sie können diese Skripte mit Hilfe eines TCP Sockets direkt an die Box senden. Alternativ können Sie auch das Programm “telnet” verwenden:

```
telnet 192.168.1.12 1233
```

Quelle: ./snippets-remote/telnet.msl

Alle Eingaben die Sie jetzt machen, werden direkt an die NetBox übertragen und bei drücken der Return Taste ausgeführt. Je nachdem, ob Sie als erstes “keep” eingeben wird die Verbindung nach Ausführung des Kommandos geschlossen - oder offen gehalten.

2.6.2.1 Eine I/O Leitung einschalten

```
io.set(1, true);
```

Quelle: ./snippets-remote/remote-gpio-on.msl

2.6.2.2 Den Zustand einer Leitung einlesen

```
print(io.get(0));
```

Quelle: ./snippets-remote/remote-gpio-in.msl

2.6.2.3 Piezo einschalten

```
mcu.piezo.beep(1000, 100); // 1kHz, 100% Vol.
```

Quelle: ./snippets-remote/remote-beep-on.msl

2.6.2.4 Piezo ausschalten

```
mcu.piezo.off();
```

Quelle: ./snippets-remote/remote-beep-off.msl

2.6.2.5 Ein kurzer Piep

```
mcu.piezo.beep(1000, 100); // 1000Hz, 100% LautstÃ€rke
pause(100); // 100ms warten
mcu.piezo.off();
```

Quelle: ./snippets-remote/remote-beep.msl

2.6.2.6 Lesen eines analogen Wertes

3 Schnittstellen

3.1 Softwareschnittstellen

Das Modul kann auf verschiedene Weise gesteuert werden:

- Steuerung per Weboberfläche
  Siehe 2.1.
- Steuerung per Java Script (AJAX)
  Dokument: **NetportWeb.pdf**.
- Steuerung mit eingebautem C Skript
  Siehe 2.3.
- Fernsteuerung mit Remote Skripten
  Siehe 2.6.
- Fernsteuerung mit- oder von einem anderen Modul (Bridge)
  Dokument: **NetportBridge.pdf**.

3.2 Hardwareschnittstellen

Schnittstelle			Voreinstellung	Dienste
Ethernet		10/100MBPs, Auto Negotiation 00:11:ef:e0:xx:xx bis 00:11:ef:e7:xx:xx	192.168.1.11	FTP, Telnet, Http, MADBridge
USB-Device		Fullspeed (12 MBits/s), Ethernet Emulation, Powered by Device	192.168.167.12	FTP, Telnet, Http, MADBridge
Stromversorgung		3,81mm, 2 Polig		

Abbildung 14: Schnittstellen

3.3 LEDs

3.3.1 Ethernet LEDs

LED	Zuordnung	Bemerkung
Rot (aussen)	Aktivität	
Grün (mittel)	Geschwindigkeit	An = 100MPS / Aus = 10MPS
Grün (innen)	Verbindung	An = FullDuplex, Aus = Half Duplex

3.3.2 Benutzer LEDs

LED	Zurdnung	Funktion
mcu.led.setLED1(BlinkTimeMS);	Rot (aussen)	Blinkt beim Booten / Stage1, Sonst Benutzer LED
mcu.led.setLED2(BlinkTimeMS);	Grün	Benutzer LED

3.4 Stromversorgung

Pin	Verwendung	Anmerkung
1	UIn	Maximal 28V
2	GND	

Tabelle 1: Steckerbelegung Stromversorgung.

Der Stromversorgungseingang ist verpolungssicher.

Hinweis

Das Modul kann sowohl über die Eingangsspannungsbuchse, als auch über **Pin 8**(UIn) und **Pin 4**(GND) versorgt werden.

3.4.1 Steckerbelegung D-SUB15

Das Modul bietet einen DSUB15 Stecker (**male**). Die Spalte “Index” bezieht sich auf die Befehler io.get(<index>, ...) und io.set(<index>, ...).

Pin	Verwendung	Index	Anmerkung
1	IN-GND		Masse (Analogteil)
2	GPIO11 / O3	11	Open Kollektor Ausgang
3	GPIO10 / O2	10	Open Kollektor Ausgang
4	GND		GND (Digitalteil), sowie Versorgung
5	GPIO9 / O1	9	Open Kollektor Ausgang
6	GPIO8 / O0	8	Open Kollektor Ausgang
7	GPIO0	0	Analogeingang
8	UIN		Versorgungsspannung (Siehe Hinweis)
9	GPIO1	1	GPIO / Analogeingang
10	GPIO2	2	GPIO / Analogeingang
11	GPIO3	3	GPIO / Analogeingang
12	GPIO4	4	GPIO / Analogeingang
13	GPIO5	5	GPIO / Analogeingang
14	GPIO6	6	GPIO / Analogeingang
15	GPIO7	7	Analogeingang

Tabelle 2: Steckerbeledung DSUB15.

Hinweis

Die Ein- und Ausgänge des PIO48OCA sind nicht explizit gegen elektrostatische Entladungen gesichert. Bitte beachten Sie beim Umgang mit dem Modul die Regeln die auch beim Umgang mit empfindlichen elektronischen Bauteilen gelten.

Hinweis

Analogmasse und Digitalmasse sind intern niederohmig verbunden. Es ist bei der externen Beschaltung sinnvoll beide Massen getrennt zu führen.

3.4.2 Pinfunktionen

Jedes I/O Pin kann per Weboberfläche, per Skript oder per Fernsteuerung, entsprechend der Pinfähigkeiten, konfiguriert werden. Zur Konfiguration gehören:

1. Name des Pins
2. Portrichtung (Ein- oder Ausgang)
3. Zusatzfunktionen (Analogeingang, SoftPWM.)

3.4.3 Name des Pins

Der Name eines I/O Pins hat rein Dokumentatorischen Wert. Er wird in der Weboberfläche angezeigt und kann frei vergeben werden.

3.4.4 Portrichtung und Zusatzfunktionen

Ein I/O Signal kann folgende Funktionen haben:

1. Analoger Eingang mit 3.3V Referenzspannung (**AIN33**)
2. Analoger Eingang mit 1.1V Referenzspannung (**AIN11**)
3. Digitaler Eingang 3.3V Pegel (**DI33**)
4. Digitaler Ausgang 3.3V Pegel (**DO33)**
5. Digitaler Openkollektorausgang (**DOOC)**
6. Invertierender digitaler Eingang 3.3V Pegel (**IDI33**)
7. Invertierender digitaler Ausgang 3.3V Pegel (**IDO33**)
8. Software implementierte PWM (Pulse Width Modulation) (**SPWM**)
9. Invertierte Software implementierte PWM (**ISPWM**)

Der Funktionsumfang variiert zwischen den Ausgängen. Folgende Tabelle zeigt die Fähigkeiten der einzelnen Pins:

Pin	AIN33	AIN11	DI33	DO33	DOOC	IDI33	IDO33	SPWM	ISPWM	Beschaltung
GPIO0	✓	✓	✗	✗	✗	✗	✗	✗	✗	Variante**2**
GPIO1	✓	✓	✓	✓	✗	✓	✓	✓	✓	Variante**1**
GPIO2	✓	✓	✓	✓	✗	✓	✓	✓	✓	Variante**1**
GPIO3	✓	✓	✓	✓	✗	✓	✓	✓	✓	Variante**1**
GPIO4	✓	✓	✓	✓	✗	✓	✓	✓	✓	Variante**1**
GPIO5	✓	✓	✓	✓	✗	✓	✓	✓	✓	Variante**1**
GPIO6	✓	✓	✓	✓	✗	✓	✓	✓	✓	Variante**1**
GPIO7	✓	✓	✗	✗	✗	✗	✗	✗	✗	Variante**2**
GPIO8	✗	✗	✗	✗	✓	✗	✗	✓	✓	Variante**3**
GPIO9	✗	✗	✗	✗	✓	✗	✗	✓	✓	Variante**3**
GPIO10	✗	✗	✗	✗	✓	✗	✗	✓	✓	Variante**3**
GPIO11	✗	✗	✗	✗	✓	✗	✗	✓	✓	Variante**3**

3.4.5 GPIO Anschlüsse Variante1

Diese Anschlüsse sind wiefolgt mit einer internen MCU verbunden. Diese Beschaltung gilt für: **GPIO1**, **GPIO2**, **GPIO3**, **GPIO4** und **GPIO5**. Bitte beachten Sie, dass für eine sinnvolle Messung analoger Spannungen ein externes Aliasingfilter beschaltet werden muss.

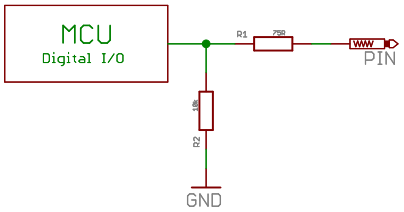


Abbildung 15: Innenschaltung Variante1.

3.4.6 Anschlüsse Variante2

Bei diesen Anschlüssen handelt es sich um analoge Eingänge. Hier ist bereits ein Aliasingfilter integriert. Diese Beschaltung gilt für **GPIO0** und **GPIO7**.

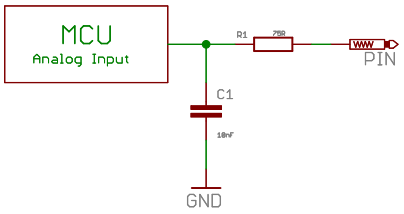


Abbildung 16: Innenschaltung Variante2.

3.4.7 Anschlüsse Variante3

Diese Beschaltung betrifft **GPIO8**, **GPIO9**, **GPIO10** und **GPIO11**. Hierbei handelt es sich um Open Collector Ausgänge, die wie folgt implementiert sind:

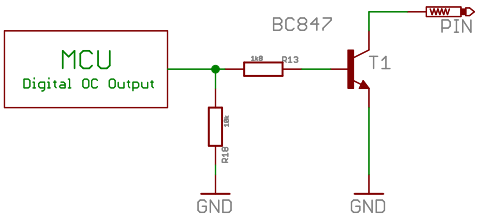


Abbildung 17: Innenschaltung Variante3.

4 Technische Daten

4.1 Elektrische Daten

Parameter	Min	Typ	Max	Einheit
Versorgungsspannung	8	12	28V	V (DC)
Leistungsaufnahme		1		W
Stromaufnahme USB		<20		µA
I_{OX}			100	mA
I_{IX}	5		20	mA
fPiezo	800		4000	Hz
Zul. Betriebstemperatur	0		60	°C
Lagertemperatur	-25		75	°C
SoftPWM Frequenz	100		10000	Hz
SoftPWM Jitter			50	µs

Tabelle 3: Elektrische Daten

4.2 Umgebungsbedingungen

Parameter	Min	Typ	Max	Einheit
Zul. Betriebstemperatur	0		60	°C
Lagertemperatur	-25		75	°C
Schutzart	-	-	-	IP20

Tabelle 4: Umgebungsbedingungen

4.3 Werkseinstellungen

Die folgenden Daten entsprechen den Daten bei Auslieferung des Moduls.

Parameter	Wert
IP-Adresse (Ethernet)	192.168.1.11
IP-Adresse (USB)	192.168.167.12

Tabelle 5: Auslieferungszustand

4.4 Software

Komponente	Wert	Version	Lizenz
First Stage Loader	LdDataFlash	svn-201	Proprietär / Conitec
Second Stage Loader	LdLinux++	svn-201	Proprietär / Conitec
Betriebssystem	Linux	2.6.34-1	GPL
Webserver	MADWeb	1.1.2.2	Proprietär
FTP Server	BusyBox	1.17.0	GPL
Basissystem	BusyBox	1.17.0	GPL
Konfiguration	dfConfig	1.7.2	Proprietär / Conitec
Script and Remote Interface	pioApp	2.0.2	Proprietär

Tabelle 6: Installierte Softwarekomponenten

4.5 Mechanische Daten

Parameter	Min	Typ	Max	Einheit
Länge		58		mm
Breite		42		mm
Höhe		18		mm
Gewicht		ca. 100g		g

Tabelle 7: Abmaße

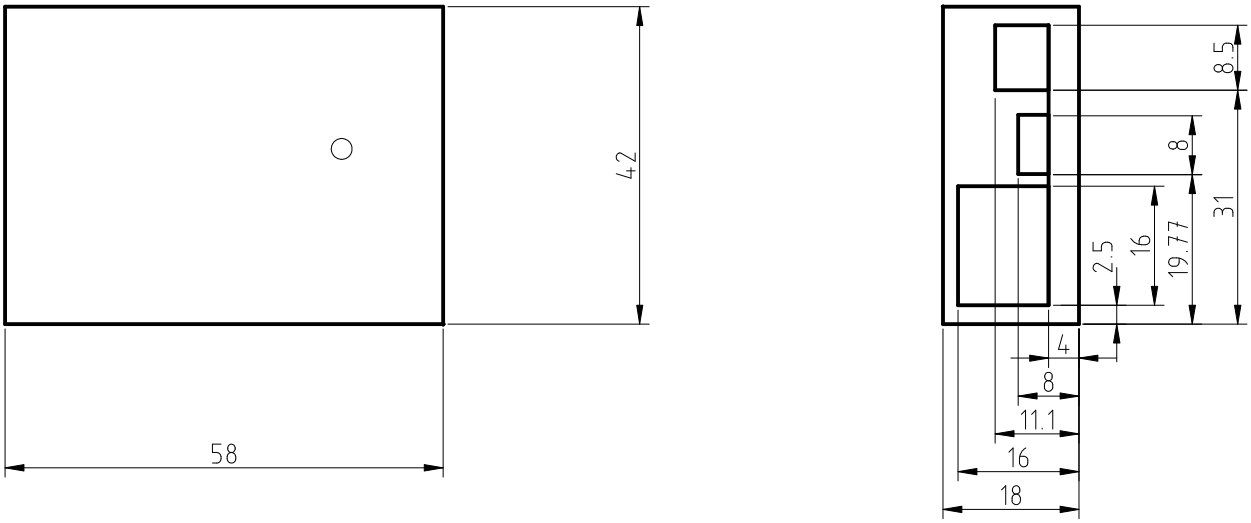


Abbildung 18: Gehäuse

5 Applikationsbeispiele

5.1 Servosteuerung

Die vier Open Kollektorausgänge (GPIO8 - GPIO11) können sehr einfach auch Modellbauservos steuern.

6 Dokument Historie

- Revision 1.
  Initial.
- Revision 1.1
  - Fixed: Some typos.
- Revision 1.2
  - Hinzugefügt: Einstellungen Sichern / Restore
  - Hinzugefügt: Fernsteuerung über Remote-Skripte
- Revision 1.3
  - Vereinfacht: Beispiele
  - Hinzugefügt: Custom Pages
- Revision 1.4
  - Hinzugefügt: LED Funktionen
  - Hinzugefügt: Mechanische Daten
- Revision 1.4a
  - Verbessert: Schreibfehler.
- Revision 1.5
  - Hinzugefügt: 2.6.2.6 - 2.6.2.8