Features

- **Modbus-Interface**
  - Widely used protocol for measuring and automation
- **gpio.NET *Core* Interface**
  - Uniform set of commands and registers
  - Supports Modbus RTU and Modbus ASCII via RS232/RS485
  - Supports Modbus ASCII via USB
  - Gateway with protocol conversion between RS232/RS485 (RTU) and USB (ASCII)
  - Configuration stored in EEPROM
- **RS232/RS485**
  - Selectable baud rate from 1200 Baud to 1MBit
  - Free configurable serial device (data width, parity, stop bits)
  - Selectable operation mode (RS232 or RS485)
- **USB**
  - USB 2.0 CDC-Device
- **Unique serial number**
  - Modbus-Address changeable through Modbus command via serial line
- **1 KByte EEPROM**
  - Seperated Config-Section and User-Section
  - Both sections can be write-protected seperately
- **4 analogue outputs**
  - Interface for voltage or current output
  - 0-5 V, 0-10 V, 0-24 mA
- **Signal generator**
  - Generates sinus, saw tooth, rectangle and triangle
  - 2,5 kHz max.

gpio.NET

AO

Preliminary

Short Description

gpio.NET
analogue Output-Card

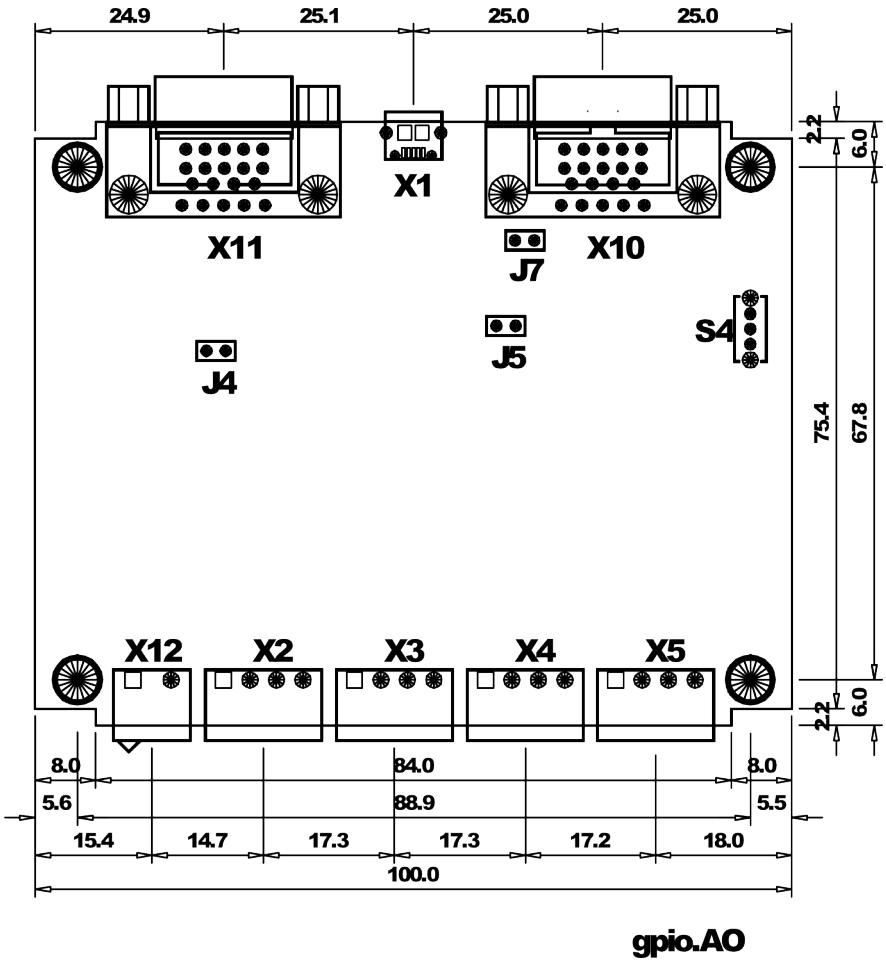
gpio.AO
U-Series

gpio.AO
R-Series

Overview

The gpio.NET *Core* interface is extended by analogue outputs and a frequency generator. Each output is capable to be either current or voltage source. The modules easily integrate into existing Modbus systems and can be identified and configured by a unique serial number even in a running system. Beyond that, they offer a uniform set of registers over the hole gpio.NET family.

Draft



Drawing 1: Dimensions

Interfaces

For communication, gpio.NET modules feature up to two Modbus interfaces. R-Series provide routing between those. The DSUB connectors (`X10` and `X11`) are only available in R-Series.

USB

The USB device port (`X1`) belongs to the standard equipment of gpio.NET cards. Because of this, the module is accessible from prevalent PC or laptops. To deal with issues resulting of this, Modbus ASCII is the only available protocol used

over USB.[1] A gpio.NET card is detected as a virtual serial port by the host system and so does not differ from a real one from a programs point of view.

USB device port is used as power source if connected to a host. As an alternative, PIN9 from either X10 or X11 can used instead.[2] When using PIN9 care has to be taken to use a well-regulated 5 volts source.

RS232

A serial interface with RS232 levels is available at gpio.NET module's X11 DSUB connector. Except the both data lines and GND no additional control lines are used.[3] The RS232 interface is capable of de- and encoding Modbus-RTU as well as Modbus-ASCII. The highest possible baud rate is 250k baud. By contemporaneous utilisation of USB and RS323, it is possible to control one additional card connected via RS232 using the hosts USB connection.

Pin	X11
1	
2	RXD
3	TXD
4	
5	GND
6	
7	
8	
9	VBUS

Table 1: Assignment RS232

RS485

The R-Series offer possibility to link via a RS485 bus. In that case, both potential data lines are available on X10 as well as on X11.[4] In this manner, a RS485 bus is easily realised with RS232 cables. As if using RS232, Modbus-RTU and Modbus-ASCII are both usable. The maximum bit rate is 1 Mbit. A 110 ohms terminating resistor is selectable by jumper J4.

Pin	X10	X11
1		
2		
3		
4	RS485+	RS485+
5	GND	GND
6	RS485-	RS485-
7		
8		
9	VBUS	VBUS

Table 2: Assignment RS485

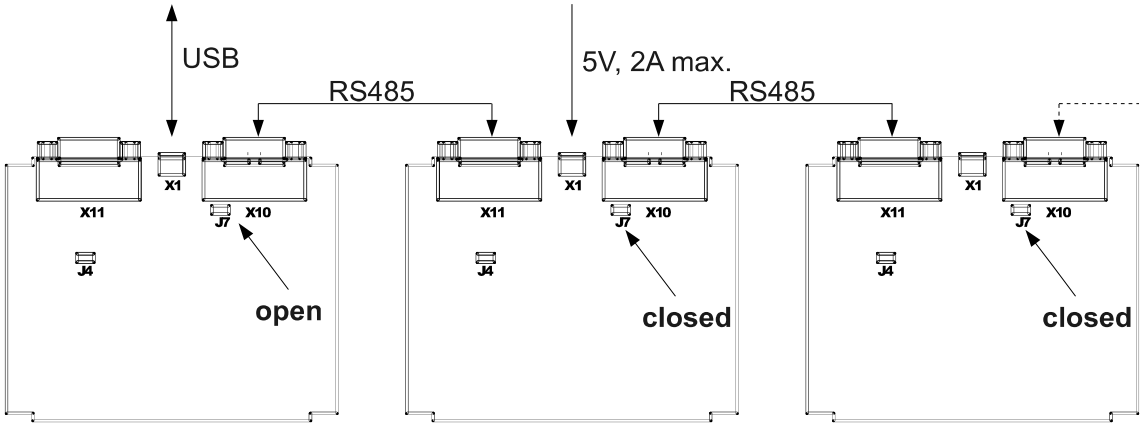
Via RS485 connected cards are routed over USB to the host, allowing access to the RS485 bus is without special PC hardware.

1 Modbus RTU – the standard mode – requires the exact temporal detection of a characters reception. This criterion cannot be guaranteed on standard PCs what causes RTU mode to become instable. Modbus ASCII is not affected by this constrain.

2 Ring indicators (PIN9) of both DSUB are connected with USB 5V power source (VBUS) (valid for X10 only with connected jumper J7). This way, power can be transmitted to the next or all gpio.NET modules in one RS485 bus system.

3 When plugging into a PC's RS232 interface, consider the RS485 assignment that is using control lines of a standard RS232 interface. These pins (DTR, DSR) should **not** be connected to a PC directly.

4 The potential data lines use standard RS232 control lines DTR and DSR. Connecting those signals to a PC directly is to be **avoided**.



Drawing 2: RS485 bus on USB with additional power supply

Individual parts of a bus constructed as shown in drawing 2 can be supplied at arbitrary points by an isolated power supply. Mind to really seperate `VBUS` of these groups. The example demonstrates this by means of the first card connected to the host PC via USB and the rest of the bus being supplied by a seperate power source. Opened jumper `J7` disrupts the `VBUS` connection on DSUB's `PIN9`.

Register description

The common set of registers simplifies working with different gpio.NET modules. Device identification, configuration of interfaces and general behaviour remain consitent throughout the family.

The gpio.Net *Core* Interface defines four sections in *Holding Register*'s address space.[5] Section `0x0000-0x00FF` contains general configuration registers – called *Core*-Register. Registers that are gpio.AO specific occupy the *Application*-Register section `0x0100-0x0FFF` which is followed by the EEPROM. Registers `0x1000-0x107F` serve as persistent configuration used at start-up. This group collects default values of *Core*- and *Application*-Registers. Section `0x2000-0x237F` is non-volatile memory available for the user.

EEPROM sections `0x1000-0x107F` and `0x2000-0x237F` can be protected against unaware writes separately.

All registers are – as common by Modbus – 16bit wide.

5 The Modbus protocol distinguishes four address ranges – *Inputregisters*, *Holdingregisters*, *Inputs* und *Coils*. *Inputs* and *Coils* offer bitwise access to ressources. While *Holdingregisters* and *Coils* are read/writeable, the other two are read-only. Each type uses a 16bit address space that can overlap each other.

0x0000	HREG_CTRL	0x1000	reserved
0x0001	HREG_PM	0x1001	HREG_PM
0x0002	HREG_BAUDSEL	0x1002	HREG_BAUDSEL
0x0003	HREG_DBITS	0x1003	HREG_DBITS
0x0004	HREG_PARITY	0x1004	HREG_PARITY
0x0005	HREG_STOP	0x1005	HREG_STOP
0x0006	HREG_SERIAL_MODE	0x1006	HREG_SERIAL_MODE
0x0007	HREG_MODBUS_MODE	0x1007	HREG_MODBUS_MODE
...	reserved	...	reserved
0x0100	CH_CFG0	0x1012	SIGNAL_FORM
0x0101	CH_CFG1	0x1013	FREQ
0x0102	CH_CFG2	0x1014	SIGNAL_CHANNEL
0x0103	CH_CFG3	...	reserved
0x0104	DATA0		
0x0105	DATA1		
0x0106	DATA2		
0x0107	DATA3		
0x0108	SIGNAL_FORM		
0x0109	SIGNAL_FREQ	0x2000	User EEPROM
0x010A	SIGNAL_CHANNEL	...	
...	reserved	0x2380	

Table 3: Holding registers

HREG_CTRL

15	14	13	12	11	10	9	8
CTRL_ACCESS_KEY (0x8E)							
7	6	5	4	3	2	1	0
reserved			CONN_ INIT	USER_ WREN	CFG_ WREN	DEFAULT	RESET

CTRL_ACCESS_KEY

Any changes to `HREG_CTRL` take only effect when the correct access key is written to the upper eight bits each time `HREG_CTRL` is accessed. This key (`0x8E`) needs to be set every time this register is about to be written to.

CONN_INIT

A logic `1` initialises the serial connection (RS232/RS485) according to the

related holding registers.

USER_WREN

Enables write access to the user EEPROM section.

CFG_WREN

Enables write access to the persitent configuration section.

DEFAULT

Restores default factory settings.

HREG_PM

This register is currently unused.

HREG_BAUDSEL

Specifies the baud rate that is used on RS232/RS485. The register contains one of those selectors described below.

Selector	Baud rate	Selector	Baud rate
0	1200	6	57600
1	2400	7	115200
2	4800	8	230400
3	9600	9	250000
4	19200	10	500000
5	38400	11	1000000

HREG_DBITS

Contains the number of data bits used on serial line – selectable between seven and eight bits.

HREG_PARITY

Configures which type of parity checking is used. Valid selectors are listed below.

Selector	Parity
0	none
1	odd
2	even

HREG_STOP

The number of additional stopbits – default is 0.

SERIAL_MODE

Selects the mode the serial line operates on. Choose 0 for RS232 or 1 for RS485.

MODBUS_MODE

To switch between MODBUS RTU and ASCII modes this register is used. Changes will not affect the USB connection that always operates in MODBUS ASCII. Choose 0 for ASCII or 1 for RTU.

CH_CFG0/1/2/3

Specifies the operating mode for the corresponding analogue output. Valid modes are listed here.

Selector	Mode
0x1000	Voltage mode 0V – 5V
0x1001	Voltage mode 0V – 10V
0x1005	Current mode 4mA – 20mA
0x1007	Current mode 0mA – 24mA

DATA0/1/2/3

The output value as positive 16bit number relative to maximum output (65535).

SIGNAL_FORM

Specifies the generated wave form.

Selector	Wave form
1	Sinus
2	Rectangle
3	Triangle
4	Saw thooth

SIGNAL_FREQ

Selects the output frequency of the signal generator. Maximum is 2500 Hz.

SIGNAL_CHANNEL

15	14	13	12	11	10	9	8
reserved							
7	6	5	4	3	2	1	0
reserved	X5 (I4/U4)	X4 (I3/U3)	X3 (I2/U2)	reserved			X2 (I1/U1)

Select the channel(s) the generated signal is directed to. Each channel corresponds to a bit configured in SIGNAL_CHANNEL. Reserved bits shall be written 0.