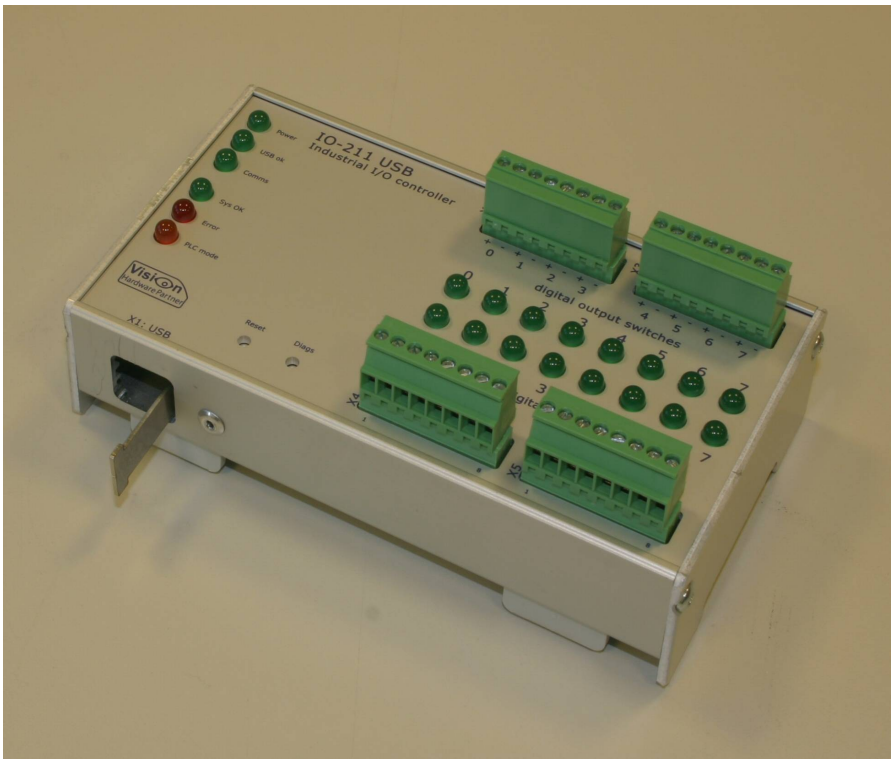
Protocol manual



USB I/O Controller

Product:	IO131, IO211
Sw Versions:	1.00 - 1.1.0
Author	Ronald Jansen

Manual code:	USB_IO_protocol_manual
Manual version:	1.0
release date:	2011-08-04

Vision Hardware Partner
Bolderweg 2
1332AT Almere
The Netherlands

Tel: +31 36 7070075
Fax: +31 36 7070045
E-mail: info@VisionHardwarePartner.nl
www: www.VisionHardwarePartner.nl

CoC (KvK): 50272772
VAT-ID: NL167571321B01
Rabobank: 1299.69.451
IBAN: NL61 RABO0129969451

1 Getting started

1.1 Connecting to the device

Two way can be used to connect to the device.

1. **The virtual comm port**
   With the virtual comm port the developer can use existing software for controlling serial ports. Communication settings like baud rate, data bits and parity do not need to be set.
2. **the USB direct drivers**
   FTDIs USB direct drivers can be used to directly find and connect the I/O controller.

1.2 protocol basics

Messages in the serial data stream are be separated by either '\n' (line feed). Or '\r' (CR).
Each message (command) to the I/O controller has to be followed by \n or \r. Each message from the USB I/O controller to the computer will be followed by both '\r' and '\n' .

<xxxx> = context defined value in hexadecimal.
<dddd> = context defined value in decimal.

For digital I/O blocks the value represents a bit mask. Each bit represents an input / output.

Commands are not case sensitive. Reports from the I/O controller are always in capitals.

1.3 Messages

The host can send commands to the USB I/O to set items, or to request status data.
The USB I/O can respond to these commands. Also it can notify the user of hardware events happening.

1.3.1 host

The host can send the following types of messages to the I/O controller:

- **get**
  get current value of an item
- **Assign**
  assign a specified value to an item
- **Notify**
  set how the USB I/O will notify the user of hardware events
- **System**
  Set / get system properties

1.3.2 controller

The USB I/O can send the following types of messages:

- **Response**
  Only sent as a reply to a message from the host. This can be a confirmation that a command was received and processed, or if the message was a data request it will provide the requested data.
- **Event notification**
  notifies the host when some external factor caused an event. This can be the change of an input, or some error condition. These events are preceded by a “!”.

1.3.3 Counter stamps

If enabled incoming event notification messages will have one ore more counter stamps attached. These can be used to link the notified event to a specific time context.

"DI=01@CT0=0000&CT1=0000"

2 Commands and events

The following paragraphs describe the possible commands, ordered by hardware functionality.

2.1 Digital inputs

Command	function	response
DIG\n	Dig In Get: Get current values of digital inputs	DI=<xxxx>\n
DIN<xxxx>\n	Dig In Notify: The 1s in the mask will be the channels from which a state change will be reported.	DIN=<xxxx>\n

Following table describes the protocol for events which can be set by the DIN command

Event	Description	Output
Digital in Change	A digital input changed, for which the event notifier was set to on.	!DI=<xxxx>\n

2.2 Digital outputs

Command	function	response
DOG\n	Dig Out Get: Get state of digital outputs.	DO=<xxxx>\n
DOA<xxxx>\n	Dig Out Assign: Assign state to digital outputs. All outputs will be set to either 1s or 0s as specified.	DOA=<xxxx>\n
DOS<xxxx>\n	Dig Out Set: Sets the outputs which correspond to the 1s in the mask to 1	DO=<xxxx>\n
DOR<xxxx>\n	Dig Out Reset: Sets the outputs which correspond to the 1s in the mask to 0	DO=<xxxx>\n

2.3 Counters

Counter inputs are connected as follows:

Device	Counter nr	Input
IO131	0	22
IO131	1	23
IO211	0	6
IO211	1	7

Both counters are 16 bit. 32 bit functionality can be added on demand by software emulation. Contact Vision Hardware Partner for details.

Following commands are used to set / read counter values:

Command	function	response
CT<d>A<xxxx>\n	Counter n Assign: Assign value to the counter specified in <d>	CT<d>=<xxxx>\n
CT<d>G\n	Counter n Getvalue: get current counter value	CT<d>=<xxxx>\n

Following commands are used to notify the user of specific counter values:

Command	function	response
CT<d>NV<xxxx>\n	Counter n Notify Value: notify computer when specified counter reaches specified value.	CT<d>NV=<xxxx>\n
CT<d>ND<xxxx>\n	Counter n Notify on Delta: Notify user when counter has counted as many steps as provided in the operand. The controller actually treats the request the same as a “notify value”. The difference is that the notify value is calculated from the provided delta value. The response will contain the calculated set value	CT<d>NV=<xxxx>\n
CT<d>NVD\n	Counter n Notify Disable: disable value set notification	CT<d>NV=D\n

Following commands are used to run a pulse on specific counter values (IO211 only):

Command	function	response
CT<d>NPE<d>\n	Selected counter will fire selected pulse. CT<d> → <d> = counter no NPE<d> → <d> = pulser no The pulser will fire at the value set with CT<d>NV	CT<d>NP=E<d>\n
CT<d>NPD\n		CT<d>NP=D\n

The following counter events can be sent by the IO211

Event	Description	Output
Counter reached value	Counter reached the value for which the notify trap was set. When the response was delayed, it will contain the most recent value.	!CT<d>=<xxxx>\n

2.4 Pulsers

The controller features a number of pulsers. These are internally timed events that can generate waveforms.

Before a pulser can be used first its waveform needs to be defined. Then after defining the waveform it can either be fired manually by issuing the “fire pulse” command, or by events like a counter reaching a preset value.

2.4.1 defining a waveform

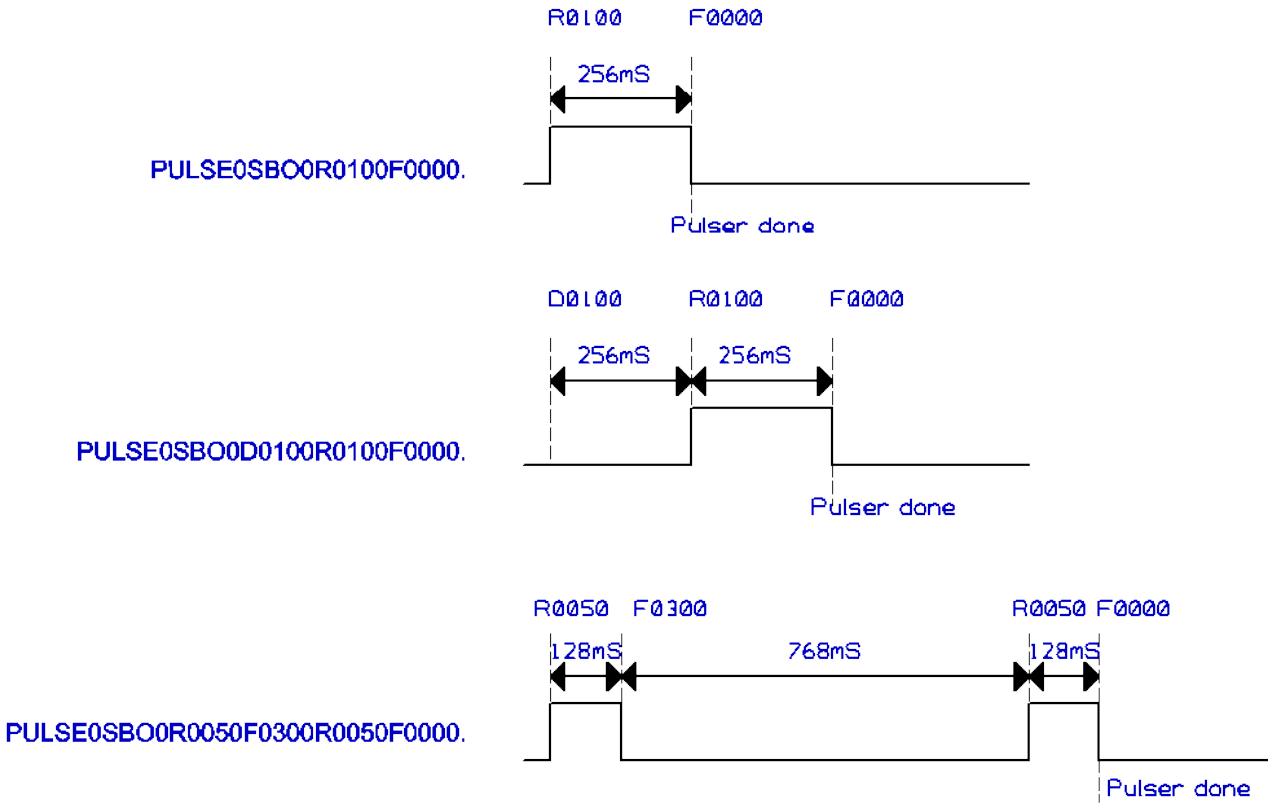
A waveform is defined using a message with the following structure:

PULSE<x>SBO<x>D<xxxx>R<xxxx>F<xxxx>.

in which:

PULSE<x>:	addresses the pulser no specified by x.
S:	Set wavefrom
BO<x>:	BO → Bank out → select output bank no specified by x. Currently only bank 0 is used.
D<xxxx>R<xxxx>F<xxxx>:	These are actually 3 separate waveform events. All in the same format. D → Delay: do nothing. Just wait the time specified in <xxxx> R → Rise: Set output to 1, then wait the time specified in <xxxx> F → Fall: Set output to 0, then wait the time specified in <xxxx> a total number of 10 waveform events can be programmed.
.	The “.” is used to indicate the end of a waveform.

Examples:



2.4.2 Firing the pulse

This can be done manually, or by setting events.

Manually:

For firing a pulse manually, the following message is sent:

PULSE<x>R → **R**un pulse no X

Fire at counter value:

This is actually a setting of the counter that is going to fire the pulse. See: counter settings.

3 System commands

3.1 Baudrate (only for IO131)

The USB I/O supports the following standard baud rates: 57600, 115200, 230400, 307200, 460800. It is also possible to set a number of (even faster) non standard baud rates. Ask Vision Hardware Partner for details on this.

The USB I/O controller powers up at the last saved baud rate. The factory setting for this is 115200bps. Other baudrates can be set using the BD command.

The BD command is used as follows:

BD<dddd>\r\n

Where <dddd> is the baudrate in bps.

The USB I/O controller will accept the new baudrate if it is available. In this case it wil respond by sending:

BD=<dddd>\r\n

The response is already sent at the new baud rate.
If the requested baudrate is not available, or the command is not understood, the USB I/O will respond by sending

?BD?\r\n

Of course the baud rate will remain the same in this case.

The currently active baud rate can be saved to non volatile memory with the following command:

BDSAVE\r\n

The USB I/O will respond to this by sending:

BD=SAVE\r\n

After this the saved baudrate will be used each time at power up.

If the saved baudrate is unknown to the user, the user might be unable to communicate with the USB I/O. The USB I/O can be reset to 115200bps by pressing the reset button next to the USB connector for longer than 4 seconds.
The USB I/O will then restart and set the baudrate to 115200bps. This is confirmed by the USB I/O sending:

!DEFAULTS\n\r

3.2 sample speed

With the command “RS\r\n” current system measuring speed can be obtained. Response will be:

RS=<dddd>\r\n

in which <dddd> is the speed in samples per second.

3.3 reboot

The command “RB\r\n” can be used for rebooting the USB I/O controller. There is no confirmation response to this. The controller will reboot right away.

3.4 Reboot source (Versions >= 1.1.0)

The controller logs the reason for a reboot. The reason for the last reset can be requested using the command:

RESSRCGET\r\n

The response will be:

RESSRC=<xx><source>\r\n

in which:

<xx> is a hexadecimally written bitmask

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		USER	JTAG	WDOG	PWRDIP	RESBTN	PWRON

<source> can be:

Reset Source	Description
PWRON	powered on normally
RESBTN	the devices reset button was pressed (on some versions this will be notified with the “USER” status
PWRDIP	controller was reset due to a dip in the supply power
WDOG	controller was reset by the watchdog
JTAG	reserved for production purposes
USER	User initiated the reset by sending RB command
CLR	in case the reset source was cleared by RESSRCCLR

If an abnormal reboot occurred this will be indicated by a blinking “error” LED.

The status can be reset by sending the command:

RESSRCCLR\r\n

The device will respond by sending:

RESSRC=00: CLR\r\n

Vision Hardware Partner
Bolderweg 2
1332AT Almere
The Netherlands

Tel: +31 36 7070075
Fax: +31 36 7070045
E-mail: info@VisionHardwarePartner.nl
www: www.VisionHardwarePartner.nl

CoC (KvK): 50272772
VAT-ID: NL167571321B01
Rabobank: 1299.69.451
IBAN: NL61 RABO0129969451

4 Error messages

4.1 Error responses

If a command was not understood, the reply will always start with '?'.

When the command is not understood at all the USB I/O controller will report: *?CMD*

If a command is partially understood. The USB I/O controller will first send '?' and then echo the request up to the point where it was not understood, followed by a '?'
Checking if a command was handled ok can be done by scanning replies for an echo of the command which was sent both with and without the '?'.

For example:

The command DIA000000 sets all inputs of a 24 input module to 0.

reply for accepted: *DIA=000000\n*
Possible error response: *?DIA?\r\n*

If the user would specify a command for digital inputs (DI) which does not exist. For example DIX, the controller would report: *?DI?\n*

4.2 Error Events

There are also some error events. These signal problems which do not directly relate to a command received by a user. These messages start with: "*!Err:*"
Following events are possible:

Error event	Description	Message
Rx buffer overflow	Rx buffer got too much data at once. Solution: Send last couple of commands again, but this time slower. (Rx buffer is 64 bytes)	!ERR:RxOVF\r\n
Tx was delayed	Happens when the controller sends large amounts of data. Either the data rate was too low to handle the flow, or the computer was not able to handle the data and requested a pause.	!ERR:TxDLY\r\n
Tx Overflow	Same as Tx delay, but now the controller decided to dump data, as it makes no sense to buffer time sensitive data.	!ERR:TxOVF\r\n

5 Custom options

A number of extra options can easily be added to the firmware. If you feel the need to have options like mentioned below contact Vision Hardware Partner to discuss the possibilities.

Vision Hardware Partner
Bolderweg 2
1332AT Almere
The Netherlands

Tel: +31 36 7070075
Fax: +31 36 7070045
E-mail: info@VisionHardwarePartner.nl
www: www.VisionHardwarePartner.nl

CoC (KvK): 50272772
VAT-ID: NL167571321B01
Rabobank: 1299.69.451
IBAN: NL61 RABO0129969451

The following extra options can be easily implemented:

- Computer watchdog reset (requires simple hardware tweak)
- generate user defined event messages on events like input change, counter at preset value, temperature exceeds set limit (requires temperature sensor module).
- Set outputs on user defined events. E.g. Set alarm ouput when life sign fails, set alarm output when temperature too high (requires temperature sensor module), set output on counter value match, reset output x msec after set output.

Vision Hardware Partner
Bolderweg 2
1332AT Almere
The Netherlands

Tel: +31 36 7070075
Fax: +31 36 7070045
E-mail: info@VisionHardwarePartner.nl
www: www.VisionHardwarePartner.nl

CoC (KvK): 50272772
VAT-ID: NL167571321B01
Rabobank: 1299.69.451
IBAN: NL61 RABO0129969451