Introduction

(https://numato.com/help/wp-content/uploads/2018/09/ZRX16_1_1200-1024x1024-1.jpeg)

Numato Lab's ZGX16/32/64 (https://numato.com/product/prodigy-zgx-series-16-32-64-channel-usb-tcp-modbus-gpio-module) and EG16/32 (https://numato.com/product/prodigy-eg-series-16-32-channel-usb-rtu-tcp-modbus-gpio-module) Channel DIN Rail compatible Modbus RTU/TCP GPIO Module with Digital IO and Analog Inputs offers great flexibility at a lower cost. Ease of use and wider operating system compatibility are the primary goals behind this product's design. Built-in USB to serial conversion allows the module to be used without any USB specific knowledge. Industry-standard Modbus protocol support allows this product to be used with most automation software that supports Modbus. For power users, this module can be controlled by writing programs in various programming languages of their choice.

Features:

- Industry-standard Modbus protocol support
- DIN Rail compatible
- 16/32/64 TTL (3.3V) compatible Digital IOs for ZGX series and 16/32 TTL (3.3V) compatible Digital IOs for EG series devices
- 8/14/32 analog inputs for ZGX series and 8/14 analog inputs for EG series boards with 12-bit resolution (multiplexed with Digital IOs)
- USB interface with CDC support. As easy as using a serial port, no USB knowledge required
- Fully isolated design with optocouplers and built-in DC-DC converter
- 12V external power supply (included)
- Can be controlled by using industry-standard automation applications or custom applications

Some of the possible uses of this module include

- Home Automation
- Lighting Control
- Garden Equipment Control
- Industrial Automation
- Test Fixtures
- DIY and Hobby

This product is compatible with the following operating systems:

- Windows XP and later versions (Windows 7, 8/8.1, 10 and future versions)
- Windows 7 Embedded and later
- Linux

- Mac OS X
- Android
- Or any other operating system that supports USB CDC devices.

And these are some of the languages and software that can be used for programming:

- C/C++
- Visual Basic (VB6, VB2008, VB2010 express and other editions)
- Visual Basic for Applications (Microsoft Office VBA)
- Perl
- Python
- JAVA
- Android
- JavaScript (Node.js)
- LabVIEW
- And many more…

ZGX series has 16/32/64 on board General purpose I/Os multiplexed with 8/14/32 Analog Inputs and EG series has 16/32 onboard General purpose I/Os multiplexed with 8/14 Analog Inputs, each connected to individual screw terminals and associated drivers capable of controlling a variety of devices Sensors, LEDs, LCD displays, etc… The module communicates with host PC over a full-speed USB link, RS485, and Ethernet interface. When connected to PC using USB/RS485, the module will appear as a serial port in Windows Device Manager (or a serial tty device in Linux and Mac).

How to Use Prodigy GPIO Modules

Using this product is very easy, thanks to supporting industry-standard Modbus protocol, the RTU and the TCP/IP network interface that allows the device to be used with most readily available Data Communication Test Software like QModMaster. This document has more information about using this device with the following software. But in no way limited to this software though.

- Windows
  - QModMaster
  - Radzio! Modbus Master Simulator
- Linux
  - Coming soon
- Mac OS X
  - Coming soon

Using this product with RTU involves the following simple steps.

1. Connect the device using a USB A to B cable or a USB to RS485 converter to the host system
2. Install driver if applicable

3. Open the COM port corresponding to the device using software that supports Modbus
4. Read/Write Coils and Registers
5. Optionally write a script or custom application to automate your task

Using this product with TCP/IP Network interface involves the following steps.

1. Connect the device using a CAT 5e Ethernet Cable(Straight through cable) to the host system
2. Connect to the IP corresponding to the device using software that supports Modbus TCP
3. Read/Write Coils and Registers
4. Optionally write a script or custom application to automate your task

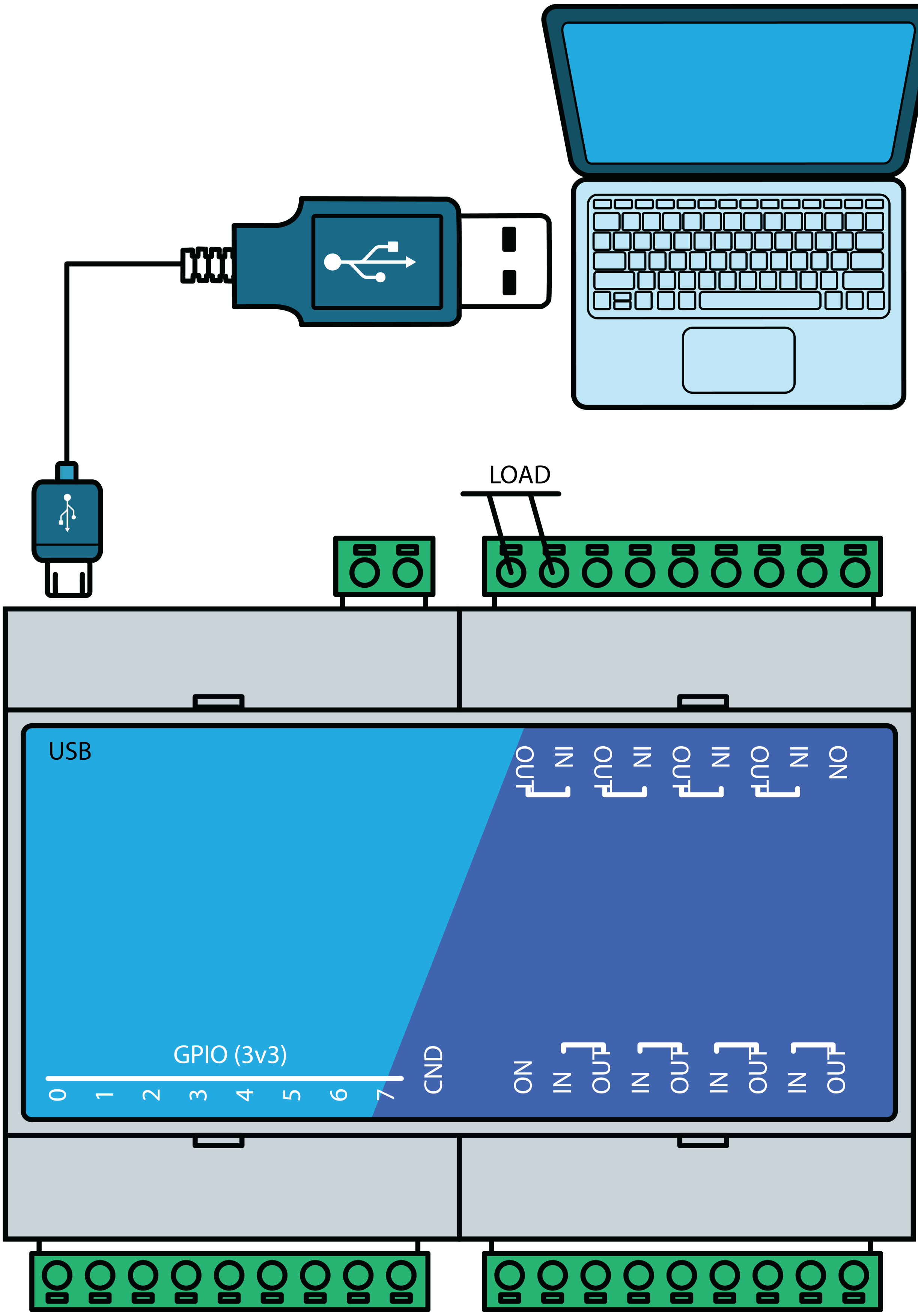
All aspects of the above steps are covered in the following sections including step by step demonstration.

Components/Tools Required

Along with your Prodigy ZGXxx/EGxx device, you will need the items in the list below for easy and fast installation.
1. USB A to B cable (Included) – For USB Interface.
2. CAT 5e Ethernet Cable(Straight through cable) – For Ethernet Interface.
3. USB to RS-485 Converter – For RS485 Interface.
4. 12V Power Supply (Included).
5. Medium size Philips screwdriver.

Connection Details

> IMPORTANT! Please exercise the utmost caution while working with electrical mains or other high voltages. Failure to comply with safety regulations may result in injury and or death.



The picture above shows a basic connection diagram that can be used in most of the situations. The connection diagram is the same for both AC and DC loads. Please make sure to use a freewheeling diode or snubber circuit if the load is inductive. More details about using inductive loads are available elsewhere in this document. Use a USB A to B cable to connect the unit to a PC. It is important to make sure that the wires used to connect loads are sufficiently rated to handle the expected load current. Exercise caution while working with high voltages. Short circuits can cause damage to the module and the PC. The following sections identify individual connections in detail.

USB Interface

The onboard full-speed USB controller helps a computer to communicate and control this module seamlessly. Use a USB A to B cable (Included) to connect the unit to a PC. The device can be connected directly to the host PC or connected through a compatible USB hub. For high-performance system integration, it is recommended to connect the device directly to one of the root ports.

RS485 Interface

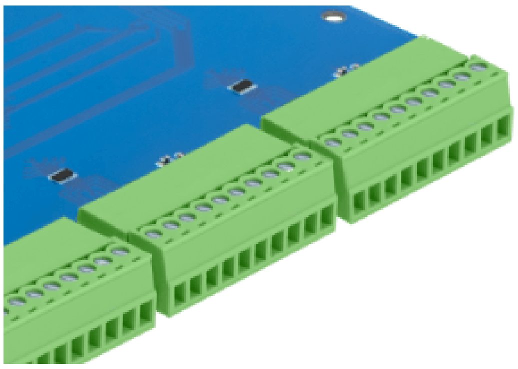
A USB to RS485 converter helps a computer to communicate and control this module seamlessly. Use a USB to RS485 converter to connect the unit to a PC. The device can be connected directly to the host PC or connected through a compatible USB hub. For high-performance system integration, it is recommended to connect the device directly to one of the root ports.

Ethernet Interface

(https://numato.com/blog/wp-content/uploads/2016/03/32-Channel-Ethernet-GPIO-Module-Ethernet-Port.jpg)The onboard Ethernet port supports Ethernet 10Mbps transmission speed that helps a computer to communicate and control this module easily. There are two basic network configurations for this board/can be used in two ways.

1. Direct connection to Local Area Network(LAN) via common straight-through Ethernet cable. Eg: Connecting the module to a switch in a network.
2. Direct connection to a PC through a cross over Ethernet cable. Eg: Connecting the module directly to the PC. Some PC/Laptops can detect and adapt to the cable type. In such situations, a straight-through cable also can be used.

GPIO/Analog Inputs



Prodigy ZGXxx device has 16/32/64 General Purpose IO pins and Prodigy EGxx device has 16/32 General Purpose IO pins that can be used for various custom applications. ZGXxx has 8/14/32 Analog Inputs and EGxx has 8/14 Analog Inputs depending on the device. All IO pins support 3.3V TTL signals and the ADC input range is 0 to +3.3V. The ADC can acquire the analog signal at the resolution of 12 bits per sample. It is recommended to use a series resistor with the GPIO/ADC pins when interfacing with other circuits. In output mode, GPIOs can source up to 20mA. So no additional circuitry is needed to drive regular LEDs. A 470 Ohms series resistor is recommended for current limiting when connecting LED to a GPIO.

In contrast to GPIOs, Analog inputs can read voltages at any level between 0 to 3.3V. It is recommended to use a series resistor to protect the input from stray voltages and spikes. The internal Analog to Digital converter supports 12 bits resolution which is adequate for most applications. The table below summarizes the GPIO and Analog to Digital Converter input positions of ZGX16/EG16 on the screw terminals.

GPIO	ADC
GPIO0	ADC0
GPIO1	ADC1
GPIO2	ADC2
GPIO3	ADC3
GPIO4	ADC4
GPIO5	ADC5
GPIO6	ADC6
GPIO7	ADC7
GPIO8	-
-	-
-	-
GPIO14	-
GPIO15	-

The table below summarizes the GPIO and Analog to Digital Converter input positions of ZGX32/EG32 on the screw terminals.

GPIO	ADC
GPIO0	ADC0
GPIO1	ADC1
GPIO2	ADC2
GPIO3	ADC3
GPIO4	ADC4
GPIO5	ADC5
GPIO6	ADC6
GPIO7	ADC7
GPIO8	ADC8
GPIO9	ADC9
GPIO10	-
GPIO11	-
GPIO12	ADC10
GPIO13	ADC11
GPIO14	ADC12
GPIO15	ADC13
GPIO16	-
GPIO17	-
-	-
-	-
-	-
GPIO29	-
GPIO30	-
GPIO31	-

And the table below summarizes the GPIO and Analog to Digital Converter input positions of ZGX64 on the screw terminals.

GPIO	ADC
GPIO0	ADC0

GPIO	ADC
GPIO1	ADC1
GPIO2	ADC2
GPIO3	ADC3
GPIO4	ADC4
GPIO5	ADC5
GPIO6	ADC6
GPIO7	ADC7
GPIO8	ADC8
GPIO9	ADC9
GPIO10	-
GPIO11	-
GPIO12	ADC10
GPIO13	ADC11
GPIO14	ADC12
GPIO15	ADC13
GPIO16	ADC14
GPIO17	ADC15
GPIO18	ADC16
GPIO19	ADC17
GPIO20	ADC18
GPIO21	ADC19
GPIO22	ADC20
GPIO23	ADC21
GPIO24	ADC22
GPIO25	ADC23
GPIO26	ADC24
GPIO27	-
-	-

GPIO	**ADC**
-	-
GPIO32	ADC25
GPIO33	ADC26
GPIO34	ADC27
GPIO35	ADC28
GPIO36	ADC29
GPIO37	ADC30
GPIO38	ADC31
GPIO39	ADC32
GPIO40	-
-	-
-	-
GPIO60	-
GPIO61	-
GPIO62	-
GPIO63	-

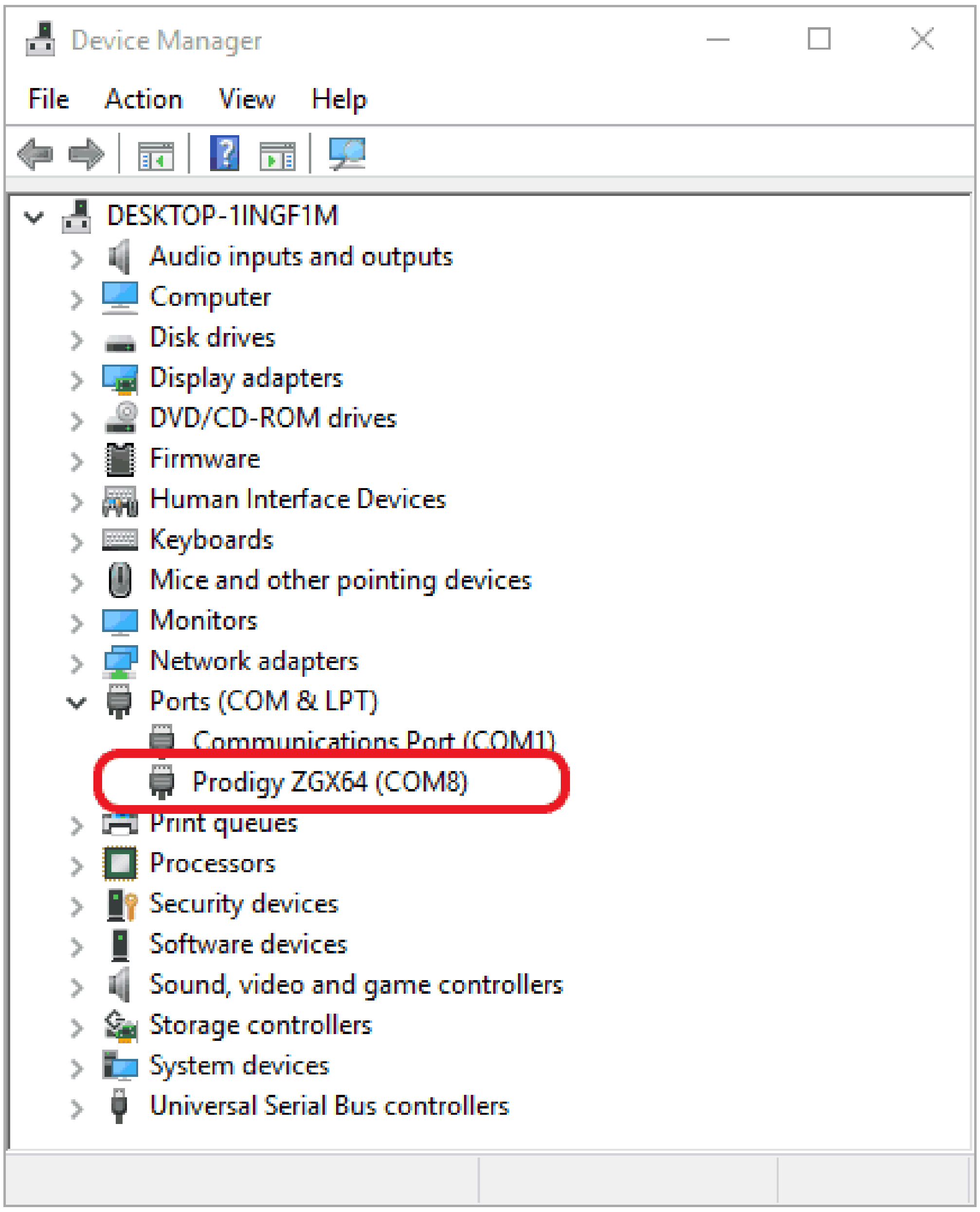
DC Power Supply

This product requires an external 12V power supply to function. **The power supply unit required is included with the product.** Connect the power supply to the connector on the product marked as +12V power. This product uses single power supply for the digital circuitry and the GPIO coils. An internal isolated DC-DC converter and a set of optocouplers ensure galvanic isolation between the digital circuitry and the GPIO coil driver circuitry.

Driver Installation

Installing Numato Lab CDC Driver - Windows Desktop and Server Editions

The driver package for Numato Lab's Prodigy models can be downloaded from the product page at https://numato.com (https://numato.com). To install the driver, unzip the contents of the downloaded driver package to a folder. Attach the USB cable to the PC and when asked by Windows device installation wizard, point to the folder where driver files are present. When driver installation is complete, the module should appear in Windows Device Manager as a serial port. The picture below shows a Prodigy ZGX64 (https://numato.com/product/prodigy-zgx-series-16-32-64-channel-usb-tcp-modbus-gpio-module) visible in Windows Device Manager. For other ZGX/EG devices, the name will be different but how the device is displayed and used is exactly the same.



Note down the name of the serial port (COM1, COM2, etc..). This information is required to control the module from the PC.

You may notice that the driver package does not come with a .sys or .exe file as most driver packages do and are expected to be that way. The driver binary necessary in this case is shipped with all copies of Windows Desktop/Server editions and gets installed automatically while

Windows is installed for the first time. The .inf and .cat files present in the driver package downloaded from http://numato.com (https://numato.com) merely associate this pre-existing driver with the attached Numato Lab device.

The following video demonstrates how to install the driver on Windows 10.

//player.vimeo.com/video/164672025?title=0&byline=0&portrait=0&color=ffffff

Installing on Windows Embedded Editions

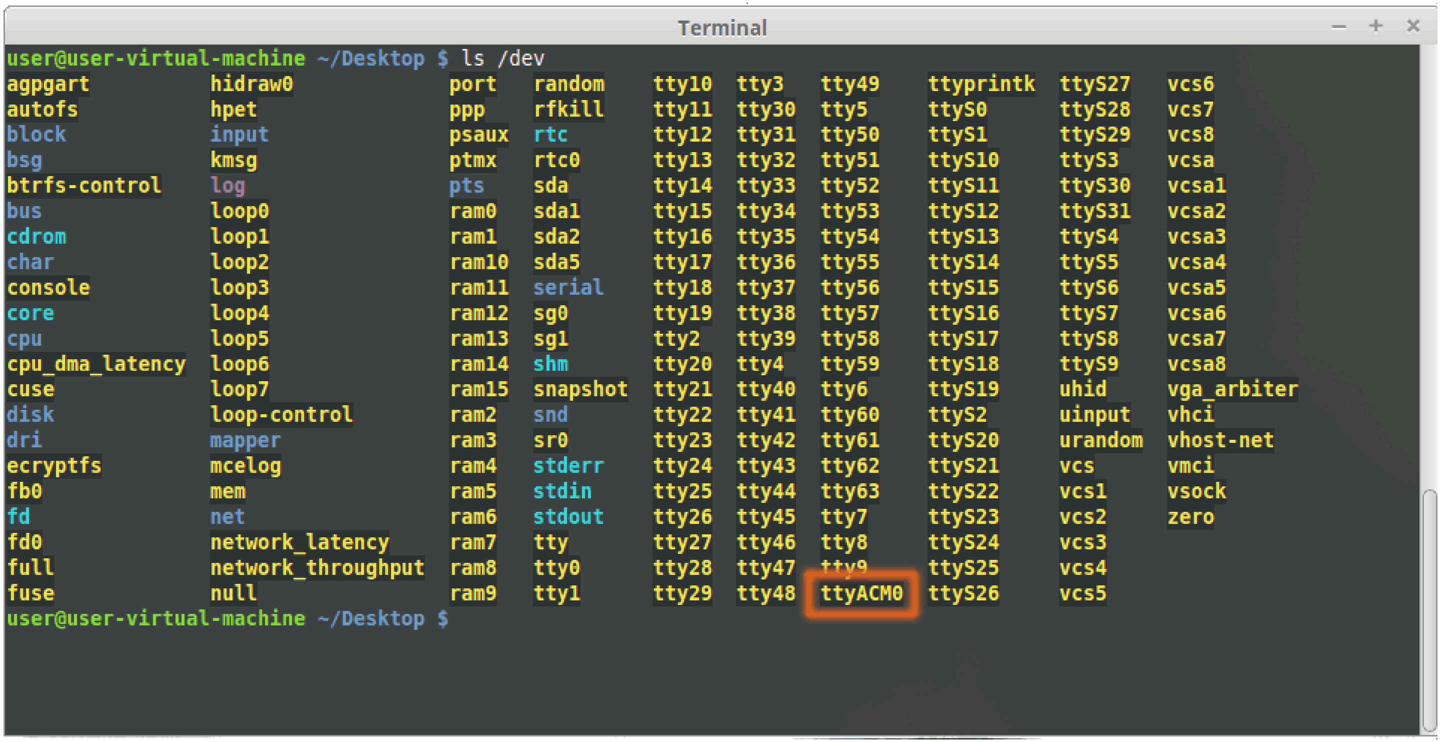
Windows Embedded editions do not install the infrastructure necessary for USB CDC by default in favor of a smaller footprint. This will cause the driver install to fail unless the necessary files are manually installed prior to installing the driver. Please follow the steps below to install the prerequisites and driver correctly. These steps are tested on Windows 7 Embedded Edition. The installation procedure may vary for other versions of Windows Embedded. Please contact Microsoft for more information.

1. Locate *winemb-inf-mdmcpq.cab* on Win 7 Embedded DVD/ISO image
2. Copy *winemb-inf-mdmcpq.cab* to a folder Ex: *C:Temp*
3. Run command *DISM.exe /online /Add-Package /PackagePath:C:Temp*
4. Wait for Windows to restart (Restart machine manually if DISM does not restart the machine automatically)
5. After the reboot is complete, plug the device to a USB port and install driver normally (Driver is available for download at the product page)

> For more information or for additional help on Windows Embedded editions, please contact Microsoft or your Windows Embedded reseller

Installing on Linux

To use any device that uses USB CDC protocol with Linux, USB CDC driver needs to be compiled into the kernel. **Fortunately, most Linux distributions (Ubuntu, Redhat, Debian, etc..) have this driver pre-installed**. The chances of you requiring to rebuild the kernel to include the USB CDC driver is very slim. When connected to a Linux machine, this product should appear as a serial port under */dev* directory. Usually, the name of the device will be *ttyACMx* or similar. The name may be different depending on the Linux distribution you have. The image below shows the result of *ls /dev* command on a Linux Mint system with a USB GPIO/Relay device attached.



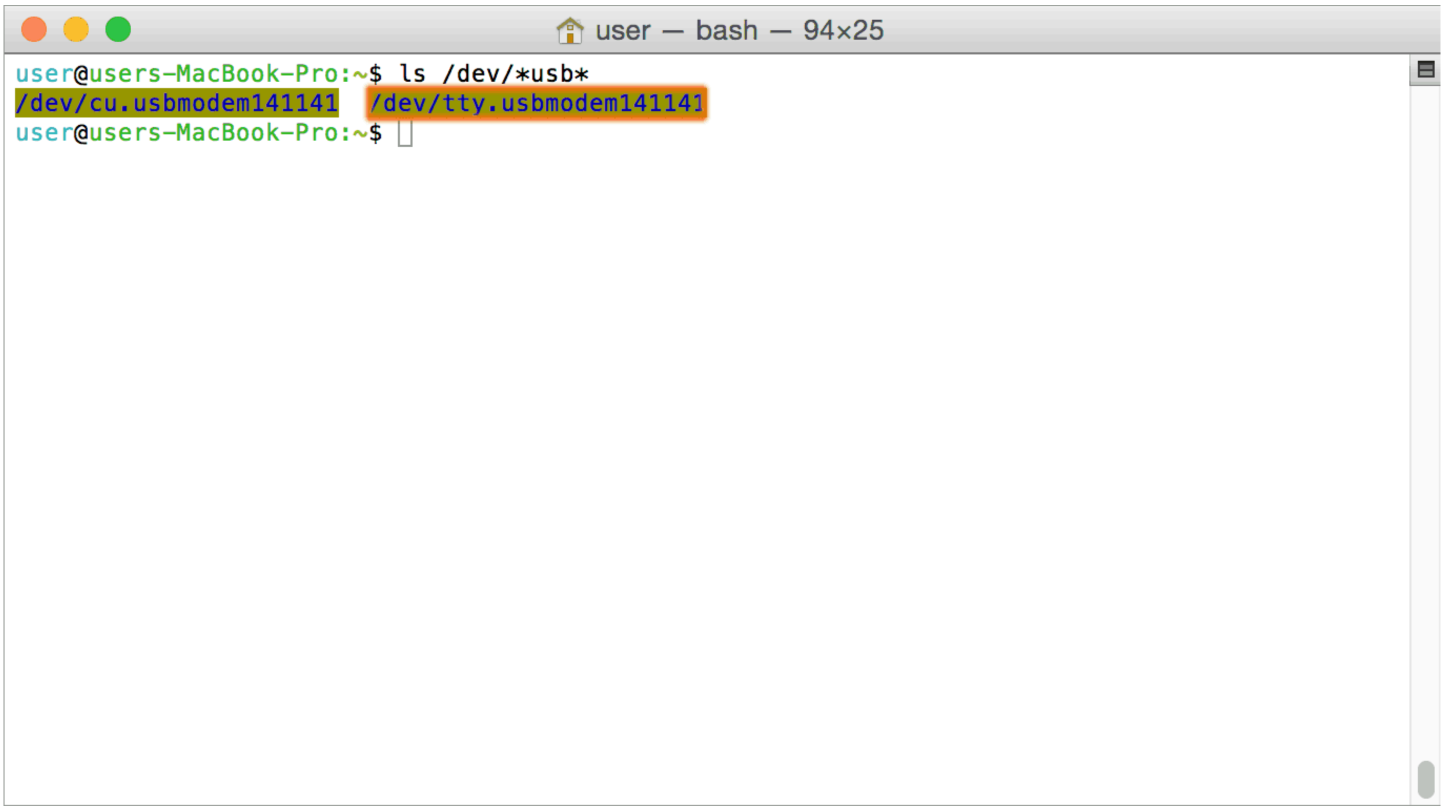
```
user@user-virtual-machine ~/Desktop $ ls /dev
agpgart          hidraw0             port   random    tty10  tty3   tty49    ttyprintk  ttyS27   vcs6
autofs           hpet                ppp    rfkill    tty11  tty30  tty5     ttyS0      ttyS28   vcs7
block            input               psaux  rtc       tty12  tty31  tty50    ttyS1      ttyS29   vcs8
bsg              kmsg                ptmx   rtc0      tty13  tty32  tty51    ttyS10     ttyS3    vcsa
btrfs-control    log                 pts    sda       tty14  tty33  tty52    ttyS11     ttyS30   vcsa1
bus              loop0               ram0   sda1      tty15  tty34  tty53    ttyS12     ttyS31   vcsa2
cdrom            loop1               ram1   sda2      tty16  tty35  tty54    ttyS13     ttyS4    vcsa3
char             loop2               ram10  sda5      tty17  tty36  tty55    ttyS14     ttyS5    vcsa4
console          loop3               ram11  serial    tty18  tty37  tty56    ttyS15     ttyS6    vcsa5
core             loop4               ram12  sg0       tty19  tty38  tty57    ttyS16     ttyS7    vcsa6
cpu              loop5               ram13  sg1       tty2   tty39  tty58    ttyS17     ttyS8    vcsa7
cpu_dma_latency  loop6               ram14  shm       tty20  tty4   tty59    ttyS18     ttyS9    vcsa8
cuse             loop7               ram15  snapshot  tty21  tty40  tty6     ttyS19     uhid     vga_arbiter
disk             loop-control        ram2   snd       tty22  tty41  tty60    ttyS2      uinput   vhci
dri              mapper              ram3   sr0       tty23  tty42  tty61    ttyS20     urandom  vhost-net
ecryptfs         mcelog              ram4   stderr    tty24  tty43  tty62    ttyS21     vcs      vmci
fb0              mem                 ram5   stdin     tty25  tty44  tty63    ttyS22     vcs1     vsock
fd               net                 ram6   stdout    tty26  tty45  tty7     ttyS23     vcs2     zero
fd0              network_latency     ram7   tty       tty27  tty46  tty8     ttyS24     vcs3
full             network_throughput  ram8   tty0      tty28  tty47  tty9     ttyS25     vcs4
fuse             null                ram9   tty1      tty29  tty48  ttyACM0  ttyS26     vcs5
user@user-virtual-machine ~/Desktop $
```

(https://docs.numato.com/wp-content/uploads/2016/01/LinuxxDeviceListing.png)

In this particular case, the device shows up as *ttyACM0* (highlighted in orange color) but it could be *ttyACM1* or *ttyACM2,* etc… depending on the specific system and other connected devices. Once the device is visible under */dev* directory, it can be treated just like any other serial device. Commands can be sent to the device using any mechanism that is valid for regular serial ports such as *screen* command or Serial Terminal Emulation applications. If there are more than one device connected to the same host computer, each device will be displayed as separate serial devices with unique names. These separate serial devices can be used to control individual devices attached.

Installing on Mac OSX

Mac OSX is usually shipped with USB CDC driver pre-installed. When connected to a Mac computer, this product should appear as a serial port under */dev* directory. Usually, the name of the device will be *tty.usbserialportx* or *tty.usbmodemx* or similar. The name may be different depending on the Mac OSX version you have. The image below shows the result of *ls /dev/*usb** command on a Mac OSX Yosemite system with a USB GPIO/Relay device attached.



```
user@users-MacBook-Pro:~$ ls /dev/*usb*
/dev/cu.usbmodem141141   /dev/tty.usbmodem141141
user@users-MacBook-Pro:~$ 
```

(https://docs.numato.com/wp-content/uploads/2016/01/ListingUSBSerialDevices.png)

In this particular case, the device shows up as *tty.usbmodem141141* (highlighted on orange color) but it could be any name starting *tty.usbmodem* or even a completely different name depending on the exact version of operating system and other connected devices. Once the device is visible under */dev* directory, it can be treated just like any other serial device. If there is more than one device connected to the same host computer, each device will be displayed as separate serial devices with unique names. These separate serial devices can be used to control individual devices attached.

The Modbus Interface

Prodigy ZGX/EG series Modbus GPIO Modules use the Modbus protocol for its primary interface. Modbus is a simple yet powerful industry-standard protocol that was originally developed by Modicon systems for transmitting/receiving information over serial links. Prodigy devices use USB/RS485 and TCP/IP as physical interfaces. Since the device uses the USB interface, it represents itself to the operating system as a classic serial device which makes Modbus a very suitable protocol for this product. This also completely hides the complexities of using USB protocol and thus making Prodigy devices as easy to use as a normal serial device.

> When working with Modbus, there are two important aspects of the protocol that the user may need to understand. 1) How to send and receive Modbus packets 2) The Modbus Register Map. One needs to learn details on the building, sending and receiving Modbus packets only if he/she is engaged in low-

level library or application development (and thus not covered in this document). For an end-user who wishes to use Prodigy devices with an off the shelf software, understanding of Modbus register map would be sufficient.

IMPORTANT! Working with single interface at a time is recommended.

Modbus Register Map

Coils

Coils are a single bit data type that represents the output state of a single bit entity such as Relay or a GPO. Please note that Coils are always used to represent an output quantity. Writing to a coil will update the output quantity with the value written. Reading a coil will return the data that was previously written. For example, writing “1” to a coil that represents a GPIO will turn ON the GPO and vice versa. Reading from a Coil that represents a turned OFF GPO will return value “0”. The Address in the table below shows the position of each coil within the Modbus register map. There are a set of 16/32/64 coils corresponding to the GPIOs on ZGX16/EG16/ZGX32/EG32/ZGX64. The data address of the Coils can be used to access the corresponding GPO.

No.	Name	Coil Number	Data Address	Size	Comments
ZGXxx/EGxx GPOs					
1	GPO0	1	0	1	GPO0 ON/OFF
2	GPO1	2	1	1	GPO1 ON/OFF
3	GPO2	3	2	1	GPO2 ON/OFF
4	GPO3	4	3	1	GPO3 ON/OFF
5	GPO4	5	4	1	GPO4 ON/OFF
6	GPO5	6	5	1	GPO5 ON/OFF
7	GPO6	7	6	1	GPO6 ON/OFF
8	GPO7	8	7	1	GPO7 ON/OFF
9	GPO8	9	8	1	GPO8 ON/OFF
10	GPO9	10	9	1	GPO9 ON/OFF
11	GPO10	11	10	1	GPO10 ON/OFF
-	-	-	-	-	-
-	-	-	-	-	-

No.	Name	Coil Number	Data Address	Size	Comments
15	GPO14	15	14	1	GPO14 ON/OFF
16	GPO15	16	15	1	GPO15 ON/OFF
ZGX32/64/EG32 GPOs Continued					
17	GPO16	17	16	1	GPO16 ON/OFF
18	GPO17	18	17	1	GPO17 ON/OFF
19	GPO18	19	18	1	GPO18/ ON/OFF
-	-	-	-	-	-
-	-	-	-	-	-
30	GPO29	30	29	1	GPO29 ON/OFF
31	GPO30	31	30	1	GPO30 ON/OFF
32	GPO31	32	31	1	GPO31 ON/OFF
ZGX64 GPOs Continued					
33	GPO32	33	32	1	GPO32 ON/OFF
34	GPO33	34	33	1	GPO33 ON/OFF
35	GPO34	35	34	1	GPO34 ON/OFF
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
61	GPO60	61	60	1	GPO60 ON/OFF
62	GPO61	62	61	1	GPO61 ON/OFF
63	GPO62	63	62	1	GPO62 ON/OFF
64	GPO63	64	63	1	GPO63 ON/OFF

Discrete Inputs

Discrete Inputs are a single bit data type that represents the input state of a single bit entity such as a GPI. For example, by reading the Input bit corresponding to a GPI, the user can get the state of the logic (HIGH/LOW) externally applied to the GPI.

No.	Name	Input Number	Data Address	Size	Comments
ZGXxx/EGxx GPIs					
1	GPI0	10001	0	1	GPI0 Status High/Low

No.	Name	Input Number	Data Address	Size	Comments
2	GPI1	10002	1	1	GPI1 Status High/Low
3	GPI2	10003	2	1	GPI2 Status High/Low
4	GPI3	10004	3	1	GPI3 Status High/Low
5	GPI4	10005	4	1	GPI4 Status High/Low
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
14	GPI13	10014	13	1	GPI13 Status High/Low
15	GPI14	10015	14	1	GPI14 Status High/Low
16	GPI15	10016	15	1	GPI15 Status High/Low
ZGX32/64/EG32 GPIs Continued					
17	GPI16	10017	16	1	GPI16 Status High/Low
18	GPI17	10018	17	1	GPI17 Status High/Low
19	GPI18	10019	18	1	GPI18 Status High/Low
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
30	GPI29	10030	29	1	GPI29 Status High/Low
31	GPI30	10031	30	1	GPI30 Status High/Low
32	GPI31	10032	31	1	GPI31 Status High/Low
ZGX64 GPIs Continued					
33	GPI32	10033	32	1	GPI32 Status High/Low
34	GPI33	10034	33	1	GPI33 Status High/Low
35	GPI34	10035	34	1	GPI34 Status High/Low
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
62	GPI61	10062	61	1	GPI61 Status High/Low

No.	Name	Input Number	Data Address	Size	Comments
63	GPI62	10063	62	1	GPI62 Status High/Low
64	GPI63	10064	63	1	GPI63 Status High/Low

Holding Registers

No.	Name	Input Number	Data Address	Size	Access	Comments
GPIO Related Registers						
1	GPO Timer Registers	40265	264	48 for ZGX16/EG16 96 for ZGX32/EG32 128 for ZGX64	WR	3*16= 24 GPO timer registers for 16 GPOs 3*32 = 96 GPO timer registers for 32 GPOs 3*64 = 192 GPO timer registers for 64 GPOs
2	GPI Power ON Pullup Value Registers	40501	500	1 for ZGX16/EG16 2 for ZGX32/EG32 4 for ZGX64	WR	Power ON Pull-up value register for GPIs
3	GPI Fail safe Pullup Value Registers	40517	516	1 for ZGX16/EG16 2 for ZGX32/EG32 4 for ZGX64	WR	Fail Safe Pull-up value register for GPIs
4	GPI Power ON Pull down Value Registers	40533	532	1 for ZGX16/EG32 2 for ZGX32/EG32 4 for ZGX64	WR	Power ON Pull down value register for GPIs
5	GPI Fail Safe Pull down Value Registers	40549	548	1 for ZGX16/EG32 2 for ZGX32/EG32 4 for ZGX64	WR	Fail safe Pull down value register for GPIs
6	GPO Power ON Open drain Value Registers	40565	564	1 for ZGX16/EG32 2 for ZGX32/EG32 4 for ZGX64	WR	Power ON Open drain value register for GPOs
7	GPO Fail safe Open drain value Registers	40581	580	1 for ZGX16/EG32 2 for ZGX32/EG32 4 for ZGX64	WR	Fail safe Open drain value register for GPOs
8	GPIO Fail safe Value Registers	40597	596	2 for ZGX16/EG16 4 for ZGX32/EG32 8 for ZGX64	WR	Fail safe value register for GPIOs

No.	Name	Input Number	Data Address	Size	Access	Comments
9	GPIO Power ON Value Registers	40605	604	2 for ZGX16/EG16 4 for ZGX32/EG32 8 for ZGX64	WR	Power ON value register for GPIOs
Watchdog Configuration Area						
10	Watchdog Holding Registers	40613	612	3	WR	Watchdog Config, timer, timeout value registers
Device Configuration Area						
11	User ID	48051	8050	2	WR	Two registers (4 bytes) for storing custom user data.
12	USB Configuration Area	48101	8100	3	WR	Slave ID, Baud rate, protocol config value registers for USB Interface
13	RS485 Configuration Area	48165	8164	2	WR	Slave ID, Baud rate value registers for RS485 Interface
14	Ethernet Configuration Area	48229	8228	23	WR & RO	Host name, IP address, MAC etc... for Ethernet Interface

Input Registers

No.	Name	Input Number	Data Address	Size	Access	Comments
Device Info Area						
1	Vendor ID	38001	8000	1	RO	
2	Product ID	38002	8001	1	RO	
3	OEM Vendor ID	38003	8002	1	RO	
4	OEM Product ID	38004	8003	1	RO	
5	HW major version and Minor version	38005	8004	1	RO	Hardware revision information.

No.	Name	Input Number	Data Address	Size	Access	Comments
6	Firmware major, minor version	38006	8005	1	RO	Firmware revision information
7	Register Map major, minor version	38009	8008	1	RO	Major and Minor versions for Modbus register map
ZGXxx Analog Inputs						
8	ADC0	30001	0	1		ADC0 Value
9	ADC	30002	1	1		ADC1 Value
10	ADC	30003	2	1		ADC2 Value
11	ADC	30004	3	1		ADC3 Value
12	ADC	30005	4	1		ADC4 Value
13	ADC	30006	5	1		ADC5 Value
14	ADC	30007	6	1		ADC6 Value
15	ADC	30008	7	1		ADC7 Value
ZGX32/ZGX64/EG32 Analog Inputs Continued						
16	ADC	30009	8	1		ADC8 Value
17	ADC	30010	9	1		ADC9 Value
18	ADC	30011	10	1		ADC10 Value
19	ADC	30012	11	1		ADC11 Value
20	ADC	30013	12	1		ADC12 Value
ZGX64 Analog Inputs Continued						
21	ADC	30014	13	1		ADC13 Value
22	ADC	30015	14	1		ADC14 Value
23	ADC	30016	15	1		ADC15 Value
24	ADC	30017	16	1		ADC16 Value
25	ADC	30018	17	1		ADC17 Value
26	ADC	30019	18	1		ADC18 Value
27	ADC	30020	19	1		ADC19 Value
28	ADC	30021	20	1		ADC20 Value

No.	Name	Input Number	Data Address	Size	Access	Comments
29	ADC	30022	21	1		ADC21 Value
30	ADC	30023	22	1		ADC22 Value
31	ADC	30024	23	1		ADC23 Value
32	ADC	30025	24	1		ADC24 Value
33	ADC	30026	25	1		ADC25 Value
34	ADC	30027	26	1		ADC26 Value
35	ADC	30028	27	1		ADC27 Value
36	ADC	30029	28	1		ADC28 Value
37	ADC	30030	29	1		ADC29 Value
38	ADC	30031	30	1		ADC30 Value
39	ADC	30032	31	1		ADC31 Value

★ RO – Read Only ★ WO – Write Only ★ WR – Write/Read

Register Formats

Let’s look at how to write values to each register.

Holding registers and its format are explained in the table below:-

No.	Register	Register Format	Comments
GPIO Timer Registers			
1	GPO Timer Value Registers	- 2*16 bit Registers for 32 bit timer value for GPOs.	Register 1 - Lower Bytes of Timer value. Default Value - 0 Minimum Value to be written - 01F4h Register 2 - Higher Bytes of Timer Value. Default Value - 0

No.	Register	Register Format	Comments
2	GPO Timer Config Register	- 16 bit register - Bit 0 - Enable Timer - Bit 1:2 - Timer Type - Bit 3:4 - Timer Action	Bit 0 :- 0 - Stops Timer 1 - Starts Timer Default Value - 0 ------------------------ Bit 1:2 :- 00 - Single shot Timer 01 - Periodic Timer Default Value - 0 ------------------------ Bit3:4 :- 00 - GPO OFF 01 - GPO ON 10 - GPO Toggle 11 - Reserved Default Value - 0
3	GPI Power ON Pull-Up Value Registers	- 16 bit Register for 16 GPIs - Each bit for each GPI	Each bit to enable/Disable pull-up for the GPIs while power ON. 0 - Pull-Up Disabled 1 - Pull- Up Enabled Default Value - 0
4	GPI Power ON Pull-down Value Registers	- 16 bit Register for 16 GPIs - Each bit for each GPI	Each bit to enable/Disable pull-down for the GPIs while Power ON. 0 - Pull-Down Disabled 1 - Pull- Down Enabled Default Value - 0
5	GPO Power ON Open Drain Value Register	- 16 bit Register for 16 GPOs - Each bit for each GPO	Each bit to enable/Disable open drain for the GPOs while Power ON. 0 - Open drain Disabled 1 - Open drain Enabled Default Value - 0
6	GPI Fail Safe Pull-Up Value Registers	- 16 bit Register for 16 GPIs - Each bit for each GPI	Each bit to enable/Disable pull-up for the GPIs while Fail safe occurs. 0 - Pull-Up Disabled 1 - Pull- Up Enabled Default Value - 0
7	GPI Fail Safe Pull-down Value Registers	- 16 bit Register for 16 GPIs - Each bit for each GPI	Each bit to enable/Disable pull-down for the GPIs while Fail safe occurs. 0 - Pull-Down Disabled 1 - Pull- Down Enabled Default Value - 0

No.	Register	Register Format	Comments
8	GPO Fail Safe Open Drain Value Register	- 16 bit Register for 16 GPOs - Each bit for each GPO	Each bit to enable/Disable open drain for the GPOs while Fail safe occurs. 0 - Open drain Disabled 1 - Open drain Enabled Default Value - 0
9	GPIO Fail Safe Value Registers	- 16 bit Register for 8 GPOs - Bit 0:7 - Each bit for each GPO - Bit 8:15 - Each bit for Input/Output mode of GPIOs	Bit 0:7 :- Each bit represents GPO status. 0 - GPO OFF 1 - GPO ON Default Value - 0 ------------------------ Bit 8:15 :- Each bit represents the Input/Output mode of corresponding GPIOs 0 - Output 1 - Input Default Value - 0
10	GPIO Power ON Value Registers	- 16 bit Register for 8 GPOs - Bit 0:7 - Each bit for each GPO - Bit 8:15 - Each bit for Input/Output mode of GPIOs	Bit 0:7 :- Each bit represents GPO status. 0 - GPO OFF 1 - GPO ON Default Value - 0 ------------------------ Bit 8:15 :- Each bit represents the Input/Output mode of corresponding GPIOs 0 - Output 1 - Input Default Value - 0
Watchdog Holding Registers			
11	Watchdog config Register	-16 bit register to Enable/Disable watchdog timer - Bit 0 - ON Bit - Bit 1 - Mode Select Bit - Bit 2:15 - Reserved	Bit 0 :- 0 - Disable Watchdog Timer 1 - Enable Watchdog Timer Default Value - 0 ------------------------ Bit 1 :- 0 - Default Mode 1 - Manual Mode(Reserved) Default Value - 0 ------------------------ Bit 2:15 :- XX - Reserved Default Value - 0

No.	Register	Register Format	Comments
12	Watchdog timer Value Register	- 16 bit Register for watchdog timer value.	Fail safe occurs when this value reaches the timeout value. Default Value - 0
13	Watchdog timeout Value	- 16 bit Register for watchdog timeout value.	Represents at what time the fail safe should occur. Default Value - 0
Common Configuration Holding Registers			
14	User ID Holding Register	- 2*16 bit registers for User ID	4 bytes for User ID. Default Value - 0
USB Configuration Holding Registers			
15	Slave ID	- 16 bit register for USB Slave ID	Default Value - 1
16	Baud Rate	- 16 bit register for USB baud rate	Baud rate can be 9600, 19200, 38400, 57600, 115200 Default Value - 57600
17	Protocol Configuration	- 16 bit register for swap between protocols(RTU,JSON,XML) - Bit 0:2 - Config Bit	Bit 0 :- 0 - Disable RTU 1 - Enable RTU Default Value - 1 Bit 1 :- 0 - Disable JSON 1 - Enable JSON Default Value - 0 ------------------------ Bit 2 :- 0 - Disable XML 1 - Enable XML Default Value - 0
RS485 Configuration Holding Registers			
18	Slave ID	- 16 bit register for RS485 Slave ID	Default Value - 1
19	Baud Rate	- 16 bit register for RS485 baud rate	Baud rates - Values ----------- --------- 9600 - 10 or 0x000A 19200 - 11 or 0x000B 38400 - 12 or 0x000C 57600 - 13 or 0x000D 115200 - 14 or 0x000E Default Value - 10 or 0x000A
Ethernet Configuration Holding Registers			
20	Host Name	- 8*16 bit registers for host name	Host name length is limited to 16 characters

No.	Register	Register Format	Comments
21	IP address	- 2*16 bit registers for IP Address	IP V4 properties of Ethernet interface.
22	IP Mask	- 2*16 bit registers for IP Mask	
23	Default Gateway	- 2*16 bit registers for Default Gateway	
24	Primary DNS	- 2*16 bit registers for Primary DNS	
25	Secondary DNS	- 2*16 bit registers for Secondary DNS	
26	DHCP Enable	-16 bit register for DHCP Enable - Bit 0 - Enable Bit - If enabled, Get IP address automatically, or static IP configuration. - Disable DHCP to write IPV4 properties.	Bit 0 :- 0 - Disable DHCP 1 - Enable DHCP Default Value - 1 ------------------------ Bit 1:15 :- X - Reserved Default Value - 0
27	TCP Port	- 16 bit registers for TCP Port	TCP port for Ethernet connection Default Value - 502
28	MAC Address	- 3*16 bit Read Only registers for MAC Address	MAC address of Ethernet interface

★ IP V4 properties of the Ethernet interface will be writable only if the DHCP is disabled.

Similarly, input registers' format is as follows:-

No.	Register	Register Format	Comments
1	Vendor ID	- 16 bit Register for Vendor ID	Default Value - 0x2A19
2	Product ID	- 16 bit Register for Product ID	ZGX64 - 0x2601 ZGX32 - 0x2602 ZGX16 - 0x2603 EG32 - 0x2101 EG16 - 0x2102 Default Value - Depends on the Product
3	OEM Vendor ID	- 16 bit register for OEM Vendor ID	Default Value - 0
4	OEM Product ID	- 16 bit register for OEM Product ID	Default Value - 0
5	HW major version and Minor version	- 16 bit register for Hardware major and minor version	Hardware revision information. Default Value - 1

No.	Register	Register Format	Comments
6	Firmware major, minor version	- 16 bit register for Firmware major and minor version	Firmware revision information. Default Value - 1
7	Firmware bugfix/patch level	- 16 bit register for Firmware bugfix/patch level	Firmware bugfix/patch level Default Value - 0
8	PNV data format major, minor version	- 16 bit register for PNV data format major, minor version	PNV data format major, minor version Default Value - 1
9	Register Map major, minor version	- 16 bit register for Register map major and minor version	Major and Minor versions for Modbus register map Default Value - 1

Controlling Prodigy ZGXxx/EGxx using off the shelf software

Prodigy ZGX/EG devices' support for Modbus protocol makes it easier to use with virtually any software that supports Modbus. This section of the document demonstrates how to use Prodigy ZGX/EG devices with some of the software that is available in the market.

Windows

qModMaster

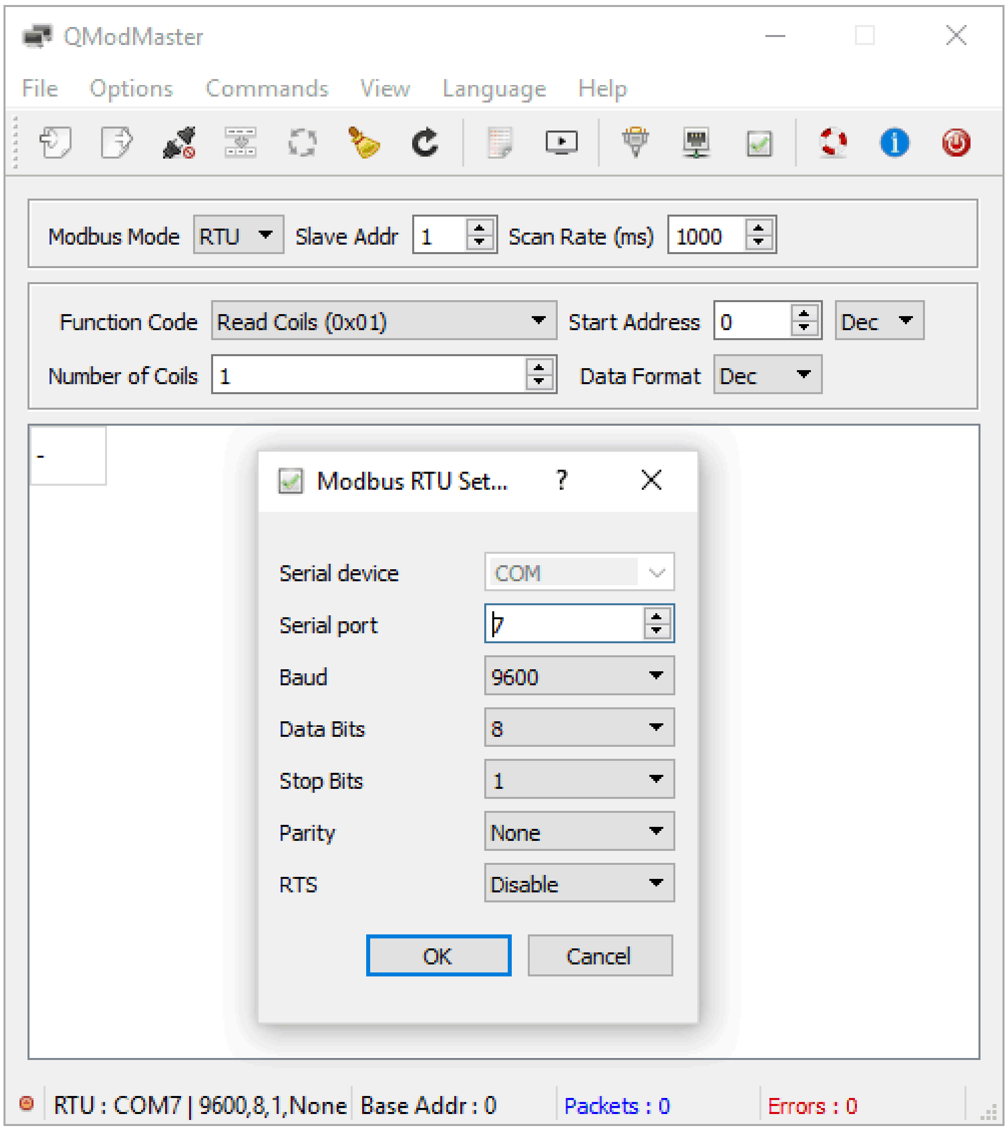
QModMaster is a free software that emulates Modbus master and can be used to access any device that is Modbus compatible. QModMaster can be downloaded for free at https://sourceforge.net/projects/qmodmaster/ (https://sourceforge.net/projects/qmodmaster/). Follow the steps below to see how to use Prodigy ZGX/EG devices with QModMaster using RTU and TCP.

Download and install QModMaster.

To control the device via Modbus RTU, follow the simple steps given below:

Step 1:

Run QModMaster and select "Modbus RTU" from the Options menu. Enter Serial Port name and other settings as in the image below and click OK. Serial Port name must match the port name assigned to the device by the Operating System.



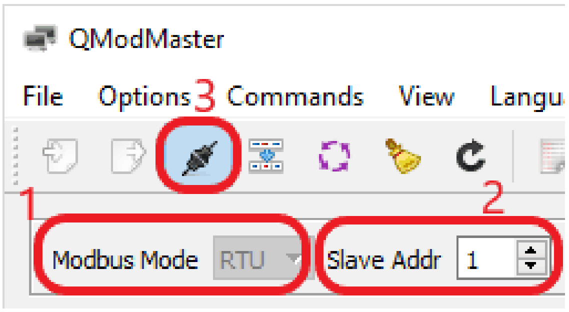
Modbus Mode RTU | Slave Addr 1 | Scan Rate (ms) 1000
Function Code Read Coils (0x01) | Start Address 0 | Dec
Number of Coils 1 | Data Format Dec

Serial device	COM
Serial port	7
Baud	9600
Data Bits	8
Stop Bits	1
Parity	None
RTS	Disable

RTU : COM7 | 9600,8,1,None Base Addr : 0 Packets : 0 Errors : 0

Step 2:

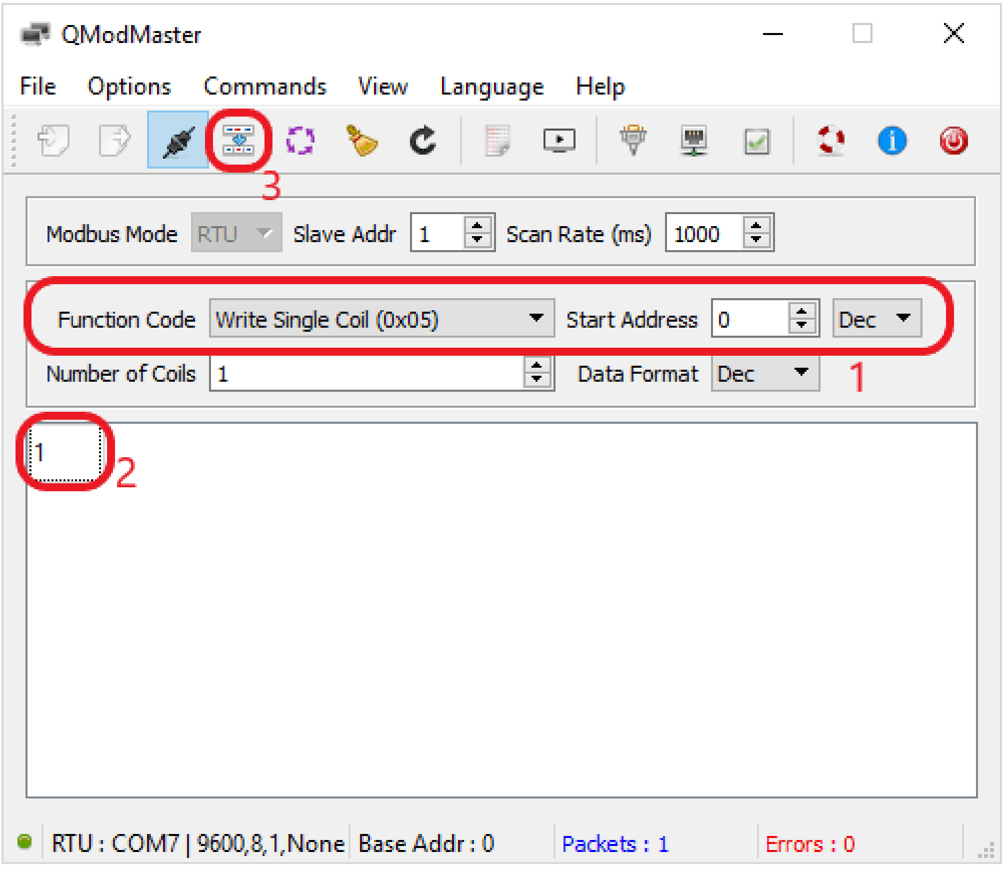
1. Select RTU as Modbus mode.
2. Select the proper Slave ID of the device.
3. Then, click the "Connect" button to connect to the device.



Step 3:

1. Select the "Write Single Coil" function code in the Function Code combo box, enter the Data Address of the first GPO (GPO Index 0) in the Start Address box.
2. Enter value 1 (corresponds to GPO High state) in the data cell.
3. Now click the Read/Write button to send the new value to the device.

If everything works fine, you will see the circuitry connected to the GPIO will be activated. To read the status of a GPO, the same sequence apply but select the "Read Coils" function code instead.

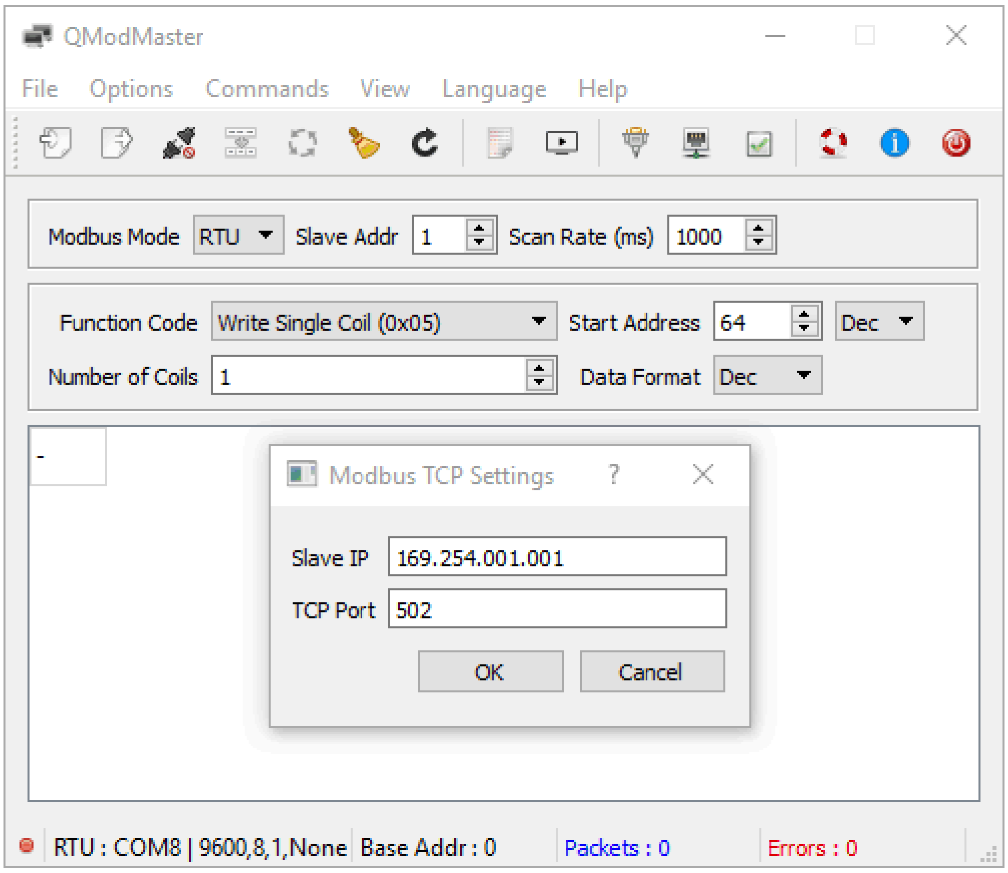


Modbus Mode RTU Slave Addr 1 Scan Rate (ms) 1000
Function Code Write Single Coil (0x05) Start Address 0 Dec
Number of Coils 1 Data Format Dec
1
RTU : COM7 | 9600,8,1,None Base Addr : 0 Packets : 1 Errors : 0

Similarly, to control the device via Modbus TCP, follow the steps below:

Step 1:

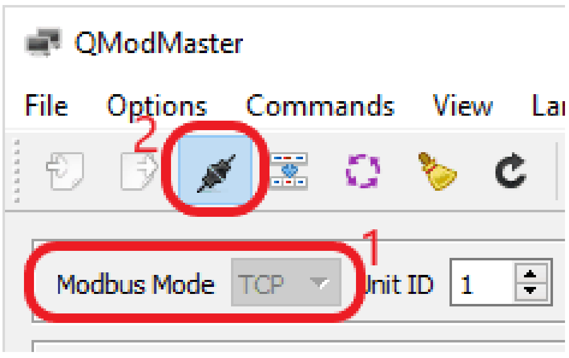
Run QModMaster and select “Modbus TCP” from the Options menu. Enter the IP Address of the device and TCP port number as in the image below and click OK.



Modbus Mode RTU Slave Addr 1 Scan Rate (ms) 1000
Function Code Write Single Coil (0x05) Start Address 64 Dec
Number of Coils 1 Data Format Dec
-
Slave IP 169.254.001.001
TCP Port 502
RTU : COM8 | 9600,8,1,None Base Addr : 0 Packets : 0 Errors : 0

Step 2:

1. Select TCP as Modbus mode.
2. Then, click the "Connect" button to connect to the device.

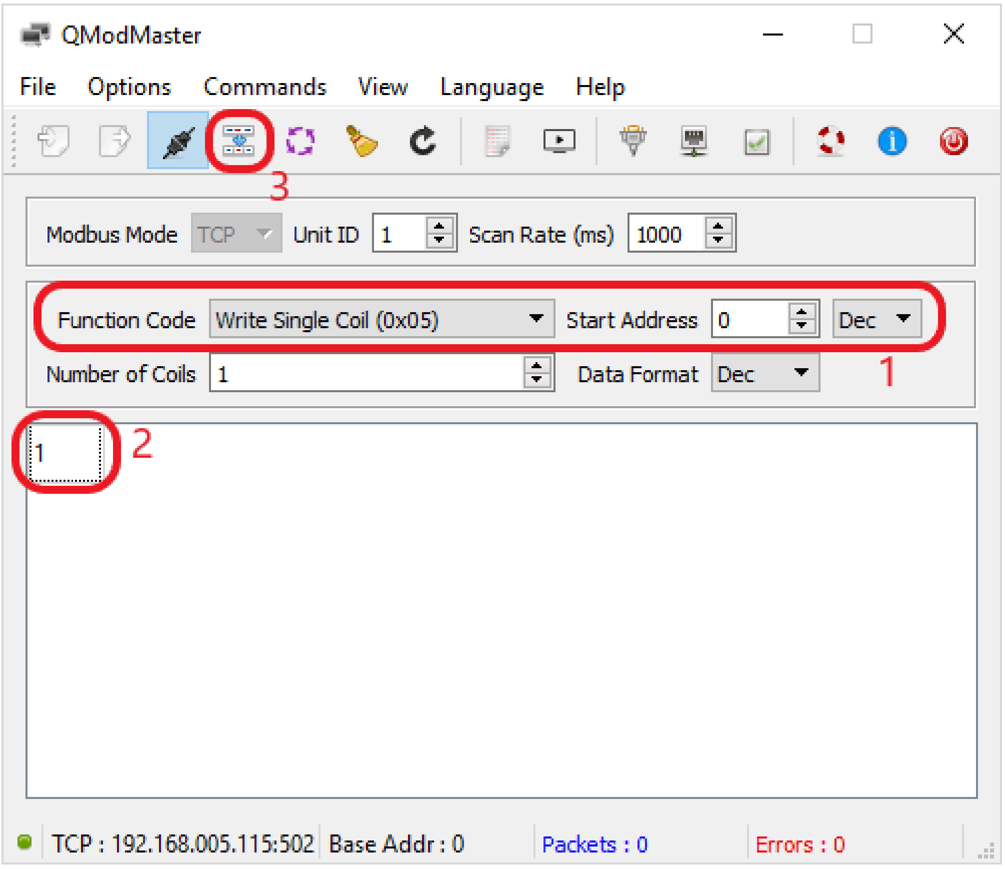


Modbus Mode: TCP

Step 3:

1. Select the "Write Single Coil" function code in the Function Code combo box, enter the Data Address of the first GPO (GPO Index 0) in the Start Address box.
2. Enter value 1 (corresponds to GPO High state) in the data cell.
3. Now click the Read/Write button to send the new value to the device.

If everything works fine, you will see the circuitry connected to the GPIO will be activated. To read the status of a GPO, the same sequence apply but select the "Read Coils" function code instead.



Modbus Mode: TCP
Unit ID: 1
Scan Rate (ms): 1000
Function Code: Write Single Coil (0x05)
Start Address: 0 Dec
Number of Coils: 1
Data Format: Dec

1

TCP : 192.168.005.115:502 | Base Addr : 0 | Packets : 0 | Errors : 0

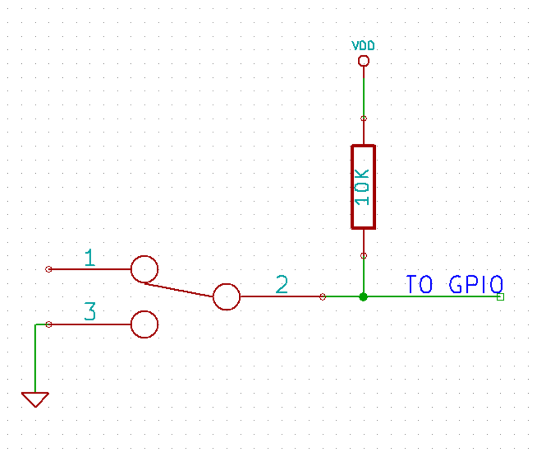
Additional Information

Analog to Digital Converters (ADCs)

Prodigy ZGX/EG devices do support Analog to Digital Conversion on some of the GPIO terminals. A list of GPIOs that supports analog function in this product is listed elsewhere in this document. GPIOs don't need to be configured especially in order to use them as analog inputs. Simply reading the Analog Input Register corresponding to the IO will automatically put the IO into Analog Input mode and read the analog value. The resolution of the ADC on this product is 12 bits. The input voltage range of the ADC is 0 – 3.3V.

GPIO with Switches

(https://numato.com/help/wp-



content/uploads/2016/01/USBGpio_Switch_connection_diagram.png)It is possible to read the position of a switch that is connected to a GPIO. An SPST or SPDT switch is recommended to use with GPIOs. Push switches that maintain the contacts closed only for a very short time so using them is discouraged. The fundamental idea of using a switch with GPIO is to have the switch cause a voltage level change at the GPIO pin when pressed. Usually, this is achieved by using an external pull-up resistor along with the switch. The pull up resistor is connected between the GPIO and VDD and the switch is connected between the GPIO and ground. When the switch is not pressed, the pull-up resistor will cause the GPIO to stay at the VDD voltage level. When the switch is pressed, the GPIO is short-circuited to the ground and stays at zero voltage. This change in voltage and thus the position of the switch can be read by simply reading the Discrete Input bit corresponding to the GPIO.

Technical Specifications

Parameter	Value	Unit
Number of GPIOs	16/32/64 for ZGX series 16/32 for EG series	

Parameter	Value	Unit
Number of analog inputs (Multiplexed with GPIOs)	8/14/32 for ZGX series 8/14 for EG series	
Power supply voltage (External)	12	V
IO Specifications		
Maximum IO source current : IO0 – IO11, IO13 – IO15, IO17 – IO19, IO25, IO26, IO28, IO29, IO31, IO32, IO36, IO37, IO40 – IO50, IO52 – IO54, IO56 – IO60, IO62, IO63	25	mA
Maximum IO source current : IO12, IO16, IO20 – IO24, IO30, IO33 – IO35, IO38, IO39, IO51	15	mA
Maximum IO source current : IO27, IO55, IO61	33	mA
Maximum IO sink current : IO0 – IO11, IO13 – IO15, IO17 – IO19, IO25, IO26, IO28, IO29, IO31, IO32, IO36, IO37, IO40 – IO50, IO52 – IO54, IO56 – IO60, IO62, IO63	25	mA
Maximum IO sink current : IO12, IO16, IO20 – IO24, IO30, IO33 – IO35, IO38, IO39, IO51	15	mA
Maximum IO sink current : IO27, IO55, IO61	33	mA
GPIO input low voltage	0.8	V
GPIO input high voltage	2	V
GPIO output low voltage	0	V
GPIO output high voltage	3.3	V
ADC Specifications		
Resolution	12	bits
Full scale range	0 – 3.3	V
Reference voltage	3.3	V
Other Information		
USB Vendor ID	0x2A19	

Parameter	Value	Unit
USB Product ID	ZGX64 - 0x2601 or 0x2604 ZGX32 - 0x2602 or 0x2605 ZGX16 - 0x2603 or 0x2606 EG32 - 0x2101 or 0x2103 EG16 - 0x2102 or 0x2104	

Frequently Asked Questions (FAQs)

Q. What are the serial parameters I need to use when communicating with this board?
A. Since this module uses USB as the underlying transport mechanism, most of the serial parameters do not affect the communication. You can leave all parameters to any legal value (Eg:9600,19200,38400 etc… for baud rate) except Flow control. Flow control needs to be set to “None”.

Q. Where do I find the driver for this product?
A. Visit http://numato.com (https://numato.com) and navigate to the product page. There will be a link to download windows driver. Linux does not require driver installation since in most cases they are shipped with the driver pre-installed.

Q. Why there is no *.sys* or *.exe* file in the Windows driver package I downloaded?
A. This product uses the USB CDC driver binary which is already present on Windows. All Windows versions (with the exception of Embedded Editions) has this driver binary installed by default. The *.inf* and *.cat* files present in the zip file helps Windows identify the device properly and associate the appropriate driver (*.sys*) to the device

Q. Does this product work with Linux?
A. Yes, this product works with Linux. Please see more details on how to use this product with Linux elsewhere in this document.

Q. Does this product work with Mac OSX?
A. Yes, this product works with Mac OSX. Please see more details on how to use this product with Mac elsewhere in this document.

Q. What is the software that this product work with?
A. This product works with almost any software that has support for standard Modbus. Some examples can be found elsewhere in this document. Different software is written by different developers with different purposes in mind. So you may encounter some software that may not work with this product. But usually, alternatives are available in most if not all cases.

Q. I’m using x language for programming. How do I find out if this language can be used to program and control the GPIO module?
A. Find out if the language of interest supports some kind of APIs/Functions/Components for serial communication. If it does, most likely you should be able to use that language with this module. It may also be possible to find libraries such as libModbus that offers high-level APIs. Using such libraries can speed up development quite a bit.

Q. I need a customized version of this product, can Numato Lab do the customization for me?
A. Yes, we can definitely do customization but there may be minimum order requirements depending on the level of customization required. Please write to help@numato.com for a quote.

Q. Where can I buy this product?
A. All Numato products can be ordered directly from our web store http://www.numato.com (http://www.numato.com). We accept major credit cards and Paypal and ship to almost all countries with a few exceptions. We do have distributors in many countries where you can place your order. Please find the current list of distributors at http://numato.com/distrib (https://numato.com/where-to-buy/).