32 CHANNEL USB GPIO MODULE

USER GUIDE

www.numato.com
sales@numato.com

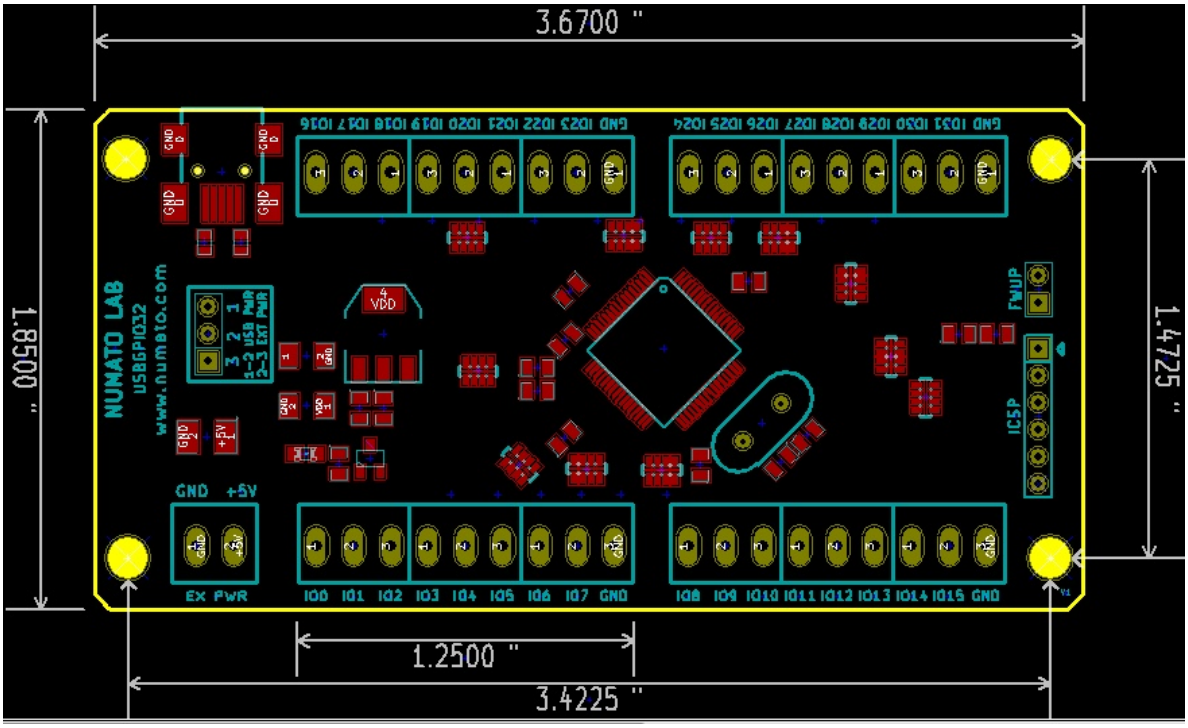
Introduction

Numato USB GPIO32 Module is a versatile piece of hardware that is useful for development as well as production purposes. The module has 32 General purpose I/Os, each connected to individual screw terminals. On board ICSP header makes it easy to be used as a development system as well. The board communicates with host PC over high-speed USB link. When connected to PC, the board will show up as a normal serial port in Device Manager. A serial terminal emulation program like HyperTerminal can be used to control individual I/Os.

Electrical Specification

System Supply Voltage	- 5V , USB powered or Externally powered
Microcontroller	- PIC18F65J50
IO Voltages	- 3.3V Max
Maximum IO source/sink	- 20ma

I/O Details and Dimensions



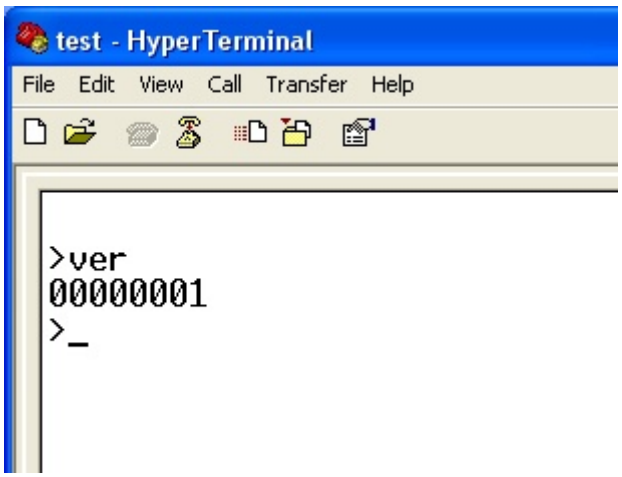
Using the Module

The primary functionality of this module is to control the GPIOs from a PC through USB link. Each I/O can be individually read or written according to your requirements. The board's digital circuitry can be powered either from USB or from an external power supply. By default the board is shipped with USB powered configuration (Jumper JP1 pins 1 and 2 connected together).

If the board needs to be powered from an external source, connect the screw terminal marked as +5V to 5V power supply and move jumper on JP1 to position 2-3. When the USB GPIO Module is connected to a PC for the first time, a window will pop up asking for the drivers. Please follow the wizard and load

the necessary driver from the 'Driver' folder in the USBGPIO32 zip package. If everything goes fine, you should be able to see a new serial port in Device manager as in the picture below (The port number can be different on different machines).

At this point, the board can be accessed by opening the corresponding serial port using a serial terminal program like HyperTerminal. When configuring the port make sure that flow control is set to "none". Rest of the settings can be left as defaults. Pressing the 'Enter' key should show a command prompt sign (>) where you can enter the commands. Enter 'ver ' to see the firmware version. The version should be printed on the console as shown in the picture right above.



See Command Reference for details of individual commands

Command Reference

ver

Command : ver
Arguments : None
Usage : ver
Example : ver
Description : Displays version of the firmware running on the board.

gpio

Command : gpio
Arguments : set/clear/read, Gpio Number
Usage : gpio set gpionum, gpio clear gpionum, gpio read gpionum
Example : gpio set 0, gpio clear 0, gpio read 0
Description : set/clear/read - Reads or writes a logic state from/to an individual GPIO pin. This command accepts GPIO number from 0 - 9 and A – V, total 32 values.
Eg: “gpio read A” reads GPIO 10, “gpio read B” reads GPIO 11 and so on.

adc

Command : adc
Arguments : read, channel
Usage : adc read channel
Example : adc read 1, adc read 2
Description : Reads voltage on the specified analog input(starting from i/o1 to i/o7).

Frequently Asked Questions

1. Can I program this board with my own firmware?

Yes, as long as you know what you are doing. You can write your own firmware and use it with this board. This board has a PIC18F65J50 microcontroller, so make sure you are building the code for this processor. Use ICSP connector to program the microcontroller.

2. Can I control the GPIOs from my own application instead of using HyperTerminal or other terminal emulators?

Yes. It is possible to control the board with different programming languages/scripts. All that we need to make sure is that the language you choose supports serial port programming. Some possible Tools/Languages are,

Visual Basic

Visual C++

Perl

Python

VBA (Word, Excel etc...)And more...

3. How fast the IOs can be switched?

This is one of the mostly asked questions. There is no particular number we can provide as an answer since the switching speed greatly depends on many factors. The fundamental USB polling rate in 10ms and theoretically this is the maximum speed that can be obtained. But in real world scenarios, it takes anywhere between 50ms – 300ms to switch the GPIOs and it depends on Operating System, other peripherals connected on the bus etc... So the maximum frequency t which the IOs