

MBAsyncServer ver: 2.0

Table of Contents

Overview:

Supporting Multiple Connections:

Data Table:

Address Ranges:

Overlaying Data Types:

Installation:

Installing MBAsyncServer:

Installing Python:

Starting MBAsyncServer:

Starting MBAsyncServer in Linux

Starting MBAsyncServer in MS Windows

Stopping MBAsyncServer

IP Port Configuration:

Remote Shutdown Command:

Command Line Options:

Using Port 502:

Standard Modbus Port:

Redirecting Port 502 on Linux:

Port 502 on Microsoft Windows:

Overview:

MBAsyncServer is a simple Modbus/TCP server with the following features:

- It can support multiple incoming connections.
 - It supports the most commonly used Modbus server functions.
 - All data is stored in a common data table.
 - The IP port used is configurable.
-

Supporting Multiple Connections:

MBAsyncServer will accept multiple incoming client Modbus/TCP connections. There is no defined limit to the number of connections that it will accept. However, if more connections result in a greater volume of communications, the increased CPU load may cause reduced server performance. Supported Modbus Functions:

The following Modbus functions are supported:

- 1 - Read Coils.
 - 2 - Read Discrete Inputs.
 - 3 - Read Holding Registers.
 - 4 - Read Input Registers.
 - 5 - Write Single Coil.
 - 6 - Write Single Holding Register.
 - 15 - Write Multiple Coils.
 - 16 - Write Multiple Holding Registers.
-

Data Table:

All Modbus functions read and write to a single unified data table. By “unified”, it is meant that all of the data types are overlaid on top of each other and occupy the same address space. This means that a single data table element may be accessed by several different functions.

Address Ranges:

The data table has the following address ranges:

- 65536 - Coils.
- 65536 - Discrete Inputs.
- 65536 - Holding Registers.
- 65536 - Input Registers.

Overlaying Data Types:

All data types are overlaid on top of each other in the following manner:

- Coils, Discrete Inputs, Holding Registers, and Input Registers are defined for their full address range (0 to 65535).
- Coils and Discrete Inputs share the same address space.
- Holding Registers, and Input Registers also share the same address space. This means that writing to a Coil also affects the corresponding Discrete Input, and writing to a Holding Register affects the corresponding Input Register.

There is also an option to pack the Coils and Discrete Inputs into the first 4096 registers (16 coils per register). When this option is enabled, writing to a Coil or Discrete Input affects the corresponding register, and visa versa. When the Coils and Discretes Inputs are *not* packed into registers, this called a separate data table. When they are packed into registers, this is called an overlaid data table. The default is 'separate'.

Example:

```
python mbasyncserver.py -d s
```

Installation:

Installing MBAsyncServer:

MBAsyncServer will come as an archive file in “tar-gz” (most platforms) or “zip” (MS-Windows) formats. The MS-Windows version has the special new-line character combination (CR-LF) required by that platform, but is otherwise identical to the standard version.

Extract the archive file into the desired location. MBAsyncServer doesn't care where you place it.

Installing Python:

This is the language support for Python which is required for MBAsyncServer to run. MBAsyncServer has been tested with version 2.5. Other versions may work but has not been tested. For Linux, Python is often already installed. If not, it can normally be installed from the distro's package manager. For MS-Windows, Python can be downloaded from <http://www.python.org>

Starting MBAsyncServer:

Starting MBAsyncServer in Linux

MBAsyncServer may be started from the command line. For Linux, the syntax is:

```
./mbasyncserver.py
```

You should see something like the following appear:

```
MBAsyncServer version 2.00
Starting server on port 8600. Fri Jan 02 18:53:36 2009
Server running...
```

Starting MBAsyncServer in MS Windows

For Microsoft Windows, the syntax is:

```
C:\Python25\python mbasyncserver.py
```

The path used for Python (e.g. “C:Python25”) may need to be changed if you have installed a different version, or have installed it in a different location.

Stopping MBAsyncServer

To shut down MBAsyncServer, press “control-C” (press the “control” and “C” keys simultaneously). This will send a signal to MBAsyncServer asking it to shut down. You should see something like the following appear:

```
Operator terminated server at Fri Jan 01 18:56:31 2009
```

IP Port Configuration:

The Modbus/TCP protocol defines IP port 502 as the “standard” port to use for Modbus/TCP communications. However, MBAsyncServer allows you to select a different IP port by means of a command line parameter “-p”. Example:

```
./mbasynserver.py -p 8502
```

Remote Shutdown Command:

As well as the normal Modbus commands, the server can also be made to respond to a special shut down signal which causes the program to exit. This is sometimes useful when testing clients, especially when running multiple servers. The shut down signal is defined by the “-q” parameter.

Example:: python mbasynserver.py -q quit.

To shut down the server remotely, simply send the shut down command to the server on the same port which it is listening to Modbus commands on, and it will exit immediately. If no shutdown command is specified, the feature is disabled by default.

Command Line Options:

The available command line options are:

- -p - port. E.g. -p 8080
 - -q - quit command. E.g. -q quit
 - -d - data table options. s = separate. o = overlay. E.g. -d s
 - -e - Print help and exit.
-

Using Port 502:

Standard Ethernet protocols use what are called “ports”. These are numbers which are part of the Ethernet packets which are used to route them to the correct application program. Both ends of the connection (client and server) have to use an agreed upon port number in order to communicate with each other. The server must “bind to the port number” (request the number from the operating system) before it can listen on that port for requests. Clients on the other hand do not have to bind to a port and are free to send requests to any port.

Standard Modbus Port:

The standard port number for Modbus/TCP is 502. It is possible to use a different port, provided both client and server can be set to use a different port number. In some cases, this isn't possible, and in most cases other cases it is most convenient to just use the standard port number.

With most operating systems, ports numbers less than 1024 are reserved for standard system services such as e-mail servers, web servers, etc. and are not available to ordinary applications. This poses a problem with Modbus/TCP. Either the Modbus/TCP server needs to gain (temporarily at least) elevated privileges, or else the incoming messages on port 502 need to be re-routed to a different port. The second choice is often the simplest, and poses the fewest security risks.

However, it is important to note that if you don't need to use the Modbus server, or if you do need it but can run it on an alternate port number, then you don't need to redirect the port. If the remote client is capable of using an alternate port, that is usually the best solution.

Redirecting Port 502 on Linux:

Linux has a built-in facility called "iptables" which can be used to block, re-route and redirect communications. Redirecting traffic arriving on port 502 to a different can easily be done by using iptables. For example, to redirect incoming traffic on port 502 to port 8502:

```
iptables -t nat -I PREROUTING -p tcp --dport 502 -j REDIRECT --to-port 8502
```

Depending upon how security is set up on your distro, you may need to either log in as "root", or (preferably) use "sudo". For example:

```
sudo iptables -t nat -I PREROUTING -p tcp --dport 502 -j REDIRECT --to-port 8502
```

To save this change permanently so that it is automatically loaded when the computer boots up (assuming you use sudo and have nano installed):

```
iptables -t nat -I PREROUTING -p tcp --dport 502 -j REDIRECT --to-port 8502
sudo sh -c "iptables-save > /etc/iptables.rules"
sudo nano /etc/network/interfaces
```

The above example ends with starting the nano editor (you can use a different editor if you wish) to edit the "interfaces" file. This file stores the Ethernet configuration. Add the following line to the end of the section for the Ethernet port which will be used for Modbus/TCP (typically "eth0"):

```
pre-up iptables-restore < /etc/iptables.rules
```

Save the configuration file and exit nano. Now Ethernet packets coming from outside the computer to port 502 will be redirected to port 8502. However, packets originating inside the same computer (e.g. 'localhost') will not be redirected. Anything originating on the same computer will need to be sent to port 8502.

Port 502 on Microsoft Windows:

Microsoft Windows does not offer any built-in security for system ports, so port 502 can be used directly without redirection. However, if you are using any server (on any port), you may need to adjust the firewall (if one is installed) to allow incoming connections.