

# Modbus Master Program

The purpose of this utility program<sup>1</sup> is to perform many of the functions of a Modbus master in a system that may include up to five Rabbit-based Modbus slave devices. This utility program is not a complete implementation of a Modbus master; it is meant to be used as a debugging tool. Double click on Modbus\ModbusMaster.exe in your Dynamic C folder to run the Modbus Master utility.

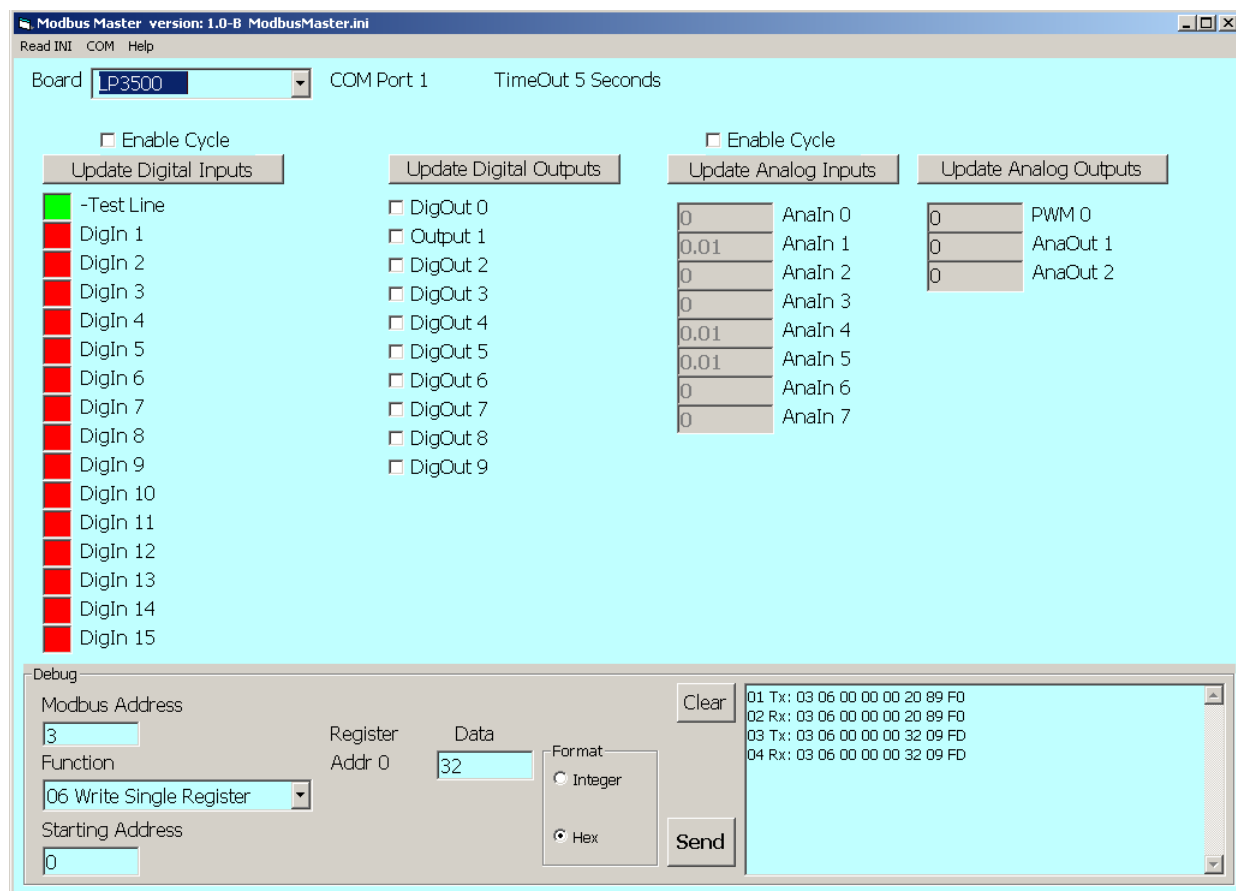
The “Debug” section of the GUI lets you see the contents of the Modbus message structure that is transmitted by the master and received from the slave. You can also send Modbus function codes explicitly using the “Debug” section.

**NOTE:** This utility does perform error checking. You can cause run-time errors by entering invalid parameters.

## Program Operation

When the program first starts up it searches for a definition file (ModbusMaster.INI) within the same folder. If one is not found, it will prompt the user to find one. [Details of the INI file](#) are discussed at the end of this document. Once an INI file has been loaded, another one may be selected by clicking on “Read INI” from the menu bar.

Figure 1. Screenshot of ModbusMaster.exe



1. The program is written in Visual Basic 6 professional. The source code can be made available on request and will be distributed as-is without support.

After the INI file has been processed, select the board with which to communicate. This is done via the Board drop down list in the upper left hand corner of the GUI. This list will display all the boards defined in the active INI file. Selecting a board will cause the GUI to display only the controls that have been defined for that board. Both the digital and analog inputs will have a grey background indicating that the states are unknown. The outputs will display the last known state. [Figure 1](#) shows the selection of the LP3500.

The menu bar item “COM” brings up a dialog box that allows you to select a new COM port and baud rate for the Modbus master.

## **Rabbit I/O Operations and ModBus Function Codes**

The program running in the target determines the I/O characteristics of that target. The Modbus library developed for the target determines the operation associated with each of the Modbus commands.

The upper section of the GUI displays the available digital and analog inputs and outputs. Clicking one of the buttons along the top causes activity in the “Debug” section of the GUI as the corresponding Modbus function code is transmitted to the slave.

Checking one of the boxes labeled “Enable Cycle” is the same as clicking the “Update ... Inputs” button every two seconds.

### **Update Digital Inputs**

Modbus function code = 0x02, “Read Discrete Inputs”

Modbus register = 0

Clicking this button requests a read of all the digital inputs. The color of the square next to the digital input indicates whether the input is off or on. Green indicates the input is grounded; red indicates the input is pulled up. If the first character of the label for a digital input is “-” then the displayed state of the input will be inverted.

### **Update Digital Outputs**

Modbus function code = 0x0F, “Write Multiple Coils”

Modbus register = 0

Clicking this button requests a write of all the digital outputs. Checking the box next to a digital output causes a “1” on that output line. On the LP3500, DigOut8 and DigOut9 are inverted by the driver; this means that checking their boxes causes a “0” on the output line.

### **Update Analog Inputs**

Modbus function code = 0x04, “Read Input Registers”

Modbus register = 3nn2

All analog inputs are read by clicking the “Update Analog Inputs” button. A single input may be updated by clicking on the name for that input.

When an analog input is requested, the system sends the command to read the channel and returns the reading in millivolts. The program will display the reading in Volts.

## Update Analog Outputs

Modbus function code = 0x06, “Write Single Register.”

Modbus register = 2nn2

All analog outputs are updated by clicking the “Update Analog Outputs” button. A single output may be updated by clicking on the name for that output.

The utility will multiply the entered value by “AnaOutMult,” which is defined in the INI file.

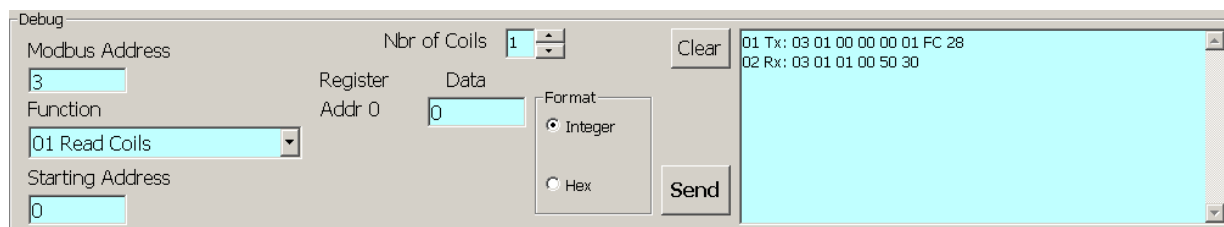
For a D/A converter it is suggested that AnaOutMult = 1000 so that the user enters the value as volts.

For PWM, it is suggested that AnaOutMult = 10 so that the user enters the value as a percentage of the desired duty cycle. In this case the valid range for entered values is 0 to 100, inclusive. This results in a close-enough-for-debugging-work approximation. For example, entering 50 causes 500 to be sent to the target, which results in close to a 50% duty cycle because 500 is close to half of the largest value allowed for the 10-bit resolution of a PWM channel.

## Debug Section of GUI

All transactions with the slave device(s) are displayed in the “Debug” section of the GUI. In addition to sending commands implicitly by clicking on one of the update buttons, you may send commands and receive responses by manually entering the appropriate Modbus function codes and parameters, then clicking “Send.”

**Figure 2. Debug Section of GUI**



### Modbus Address

The text box labeled “Modbus Address” refers to the Modbus address of the slave. This number is user-defined; the valid range is 1 to 247, inclusive. The address “0” is reserved for broadcast and is not supported in the Rabbit implementation of Modbus. In the sample program `Modbus_Serial_Slave.c`, the Modbus address is defined in the macro `MY_MODBUS_ADDRESS`. The INI file also defines this slave identifier, using the parameter “Address.” These two numbers must match for Modbus communication to occur.

### Function

The dropdown menu labeled “Function” is where you select the Modbus function code to transmit to the Modbus slave. Some functions allow you to read or write multiple controls. A scroll box will appear when appropriate that allows you to select the number of controls to read or write. In [Figure 2](#) the scroll box is labeled “Nbr of Coils” to coordinate with the “Read Coils” function selected in the dropdown menu. The supported Modbus function codes are listed below:

- 01 Read Coils - Read a maximum of 32 contiguous coils. “Coil” is a Modbus term for discrete output, also known as a digital output.
- 02 Read Discrete Inputs - Read a maximum of 32 digital inputs.

- 03 Read Holding Registers - Read a maximum of 4 analog outputs.
- 04 Read Input Registers - Read a maximum of 4 analog inputs.
- 05 Write Single Coil - Write a value of “0” to turn the digital output off or “1” to turn it on. Values greater than “1” will be evaluated as “1” or will cause a run-time error.
- 06 Write Single Register - Write an analog output.
- 15 Write Multiple Coils - Write a maximum of 32 digital outputs.
- 16 Write Multiple Registers - Write a maximum of 4 analog outputs.

### **Starting Address**

The text box labeled “Starting Address” tells the slave where to start the read or write operation.

### **Data**

The text box labeled “Data” allows you to enter values to write to the slave and also displays values that have been read from the slave. The radio buttons next to the “Data” text box allow you to tell the utility whether you want a “Data” value to be interpreted as integer or hexadecimal. For example, if you enter “10” and the “Integer” radio button is selected, the value will be interpreted as ten and not sixteen when you click the “Send” button.

### **Transmit/Receive Window**

This window displays the byte values of the Modbus messages. A transaction number is prepended to each message to make it easier to discern message sequences. This value will increment from 1 to 99 then start over. To interpret the byte values, you will need to refer to a Modbus specification.

### **INI File Format Details**

The purpose of the INI file is to define the I/O characteristics of the various Modbus slaves devices connected to the Modbus master. It is a text file with a series of parameter names and values. The format of the file is somewhat fixed and there is little to no error checking when it is parsed. A sample file can be found at the end of this document. The default file name is `ModbusMaster.INI`.

The INI file has two sections: global definitions and board definitions

#### **The global definitions are:**

- Comm Port - specify the PC comm port
- Baud Rate - and its baud rate
- CycleTime - number of seconds between automatic refreshes

#### **The board specific definitions are:**

- Board - start the values for a new Modbus slave device; each section that defines a board must start with this parameter
- Address - the Modbus address of the Modbus slave in the range 1 to 247, inclusive
- Name - a name for the board
- RespTime - the time to wait, in seconds, before a timeout error occurs
- DigInCount - the number of digital inputs, maximum of 32
- DigOutCount - the number of digital outputs, maximum of 32

- AnaInCount - the number of analog inputs, maximum of 10
- AnaOutCount - the number of analog outputs, maximum of 10
- AnaOutMult - the multiplier value for the analog outputs
- DigIn\_nn - name for Digital Input nn. If the first character of the name is a “-” the displayed state of the input will be inverted.
- DigOut\_nn - name for Digital Output nn. If the pin is capable of input or output and it is being used as an input then the name should be set to “input.” This will cause the control to be disabled as an output.
- AnaIn\_nn - name for Analog Input nn
- AnaOut\_nn - name for Analog Output nn

All “name” parameter are limited to 15 characters.

The INI file shown here has the parameters for each board indented with spaces for readability. Do NOT use the tab character as the parser does not handle them. A comment is allowed on a line by itself or at the end of a line and in both cases is initiated by a semi-colon.

As you can see by comparing the contents of this INI file with the GUI in [Figure 1](#), the labels for each of the controls are determined by the names in the active INI file. For example, the label for the first analog output on an LP3500 is “PWM 0.” The other two analog outputs are not explicitly named, so their labels appear as the default names “AnaOut 1” and “AnaOut 2.”

**NOTE:** There is no provision in this utility to set the direction of a configurable I/O pin. For digital I/O pins that can be either input or output, the program running on the target board must define the operation of the pins in the way described in the board’s user manual.

```

;INI file for use with ModBusMaster.exe
; Parameters for the LP3500 and the BL2600

Comm Port = 1      ; comment on end of line
Baud Rate = 9600
Board
  Address = 3
  Name = LP3500
  RespTime = 5
  DigInCount = 16
  DigOutCount = 10
  AnaInCount = 8
  AnaOutCount = 3
  AnaOutMult = 10
  AnaOut_00 = PWM 0
  DigIn_00 = -Test Line
  DigOut_01 = Output 1
Board
  Address = 2
  Name = BL2600
  RespTime = 1.0
  DigInCount = 32
  DigIn_04 = output 4
  DigIn_05 = output 5
  DigIn_06 = output 6
  DigIn_07 = output 7
  DigOutCount = 20
  DigOut_00 = input
  DigOut_01 = input
  DigOut_02 = input
  DigOut_03 = input
  DigOut_16 = HiCur 0
  DigOut_17 = HiCur 1
  DigOut_18 = HiCur 2
  DigOut_19 = HiCur 3
  AnaInCount = 8
  AnaOutCount = 4

```