Communication Add-on

Modbus - RTU

Addendum A

Revision History:
>   1.00 – 1st Release
>   1.01 – Added RSM
>   1.02 – Updated bookmarks

Table of Contents

1. Introduction

The Modbus-RTU protocol runs over the RS-485 communication medium. For tips regarding setting up a RS-485 network, see the user manual specific to your controller.

The Arcus implementation of Modbus-RTU supports the holding register data type only. Coils, Discrete Inputs and Input Registers are not supported.

2. Communication Settings

Baud Rate

The RS-485 baud rate can be changed using the **DB** command.

Please note that the device baud rate must be within the range [1, 5].

Device Baud Value	Baud Rate (bps)
1	9600
2	19200
3	38400
4	57600
5	115200

To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the new baud rate will be written to memory. Note that before a power cycle is completed, the settings will not take effect.

By default: Baud rate is set to: **1 (9600 bps)**

This setting can be made either through RS-485 ASCII or USB communication depending on your controller.

Modbus Device Number

Arcus Modbus controllers allow the user to set the Modbus device number (used for addressing) from the range of [1-127]. In order to make this change, use the **DNM** command.

To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the new device number will be written to memory. Note that before a power cycle is completed, the settings will not take effect.

By default: Modbus device number is set to **1**

This setting can be made either through RS-485 ASCII or USB communication depending on your controller.

RS-485 Communication Mode

Changing from RS-485 to Modbus
By default, Arcus controllers are configuring for ASCII protocol when using RS-485 communication. In order to enable Modbus-RTU communication, set **RSM=1**.

To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the communication mode setting will be written to memory. Note that before a power cycle is completed, the settings will not take effect.
This setting must be made through RS-485 ASCII.

Below is a table of the ASCII commands that are mentioned above. To see a complete list of all ASCII commands, see the user manual specific to your controller.

Command	Description	Return
DB	Return the current baud rate of the device	1 – 9600 bps 2 – 19200 bps 3 – 38400 bps 4 – 57600 bps 5 – 115200 bps
DB=[1,2,3,4,5]	Set the baud rate of the device	OK
DNM	Get Modbus device number	[1-127]
DNM=[1-127]	Set Modbus device number	OK
RSM	Get RS-485 Communication mode	0 – ASCII 1 – Modbus-RTU
RSM=[0,1]	Set RS-485 Communication mode	OK
STORE	Store settings to flash	OK

Changing from Modbus to RS-485
Once your controller is in Modbus communication, it can no longer accept ASCII commands. In order to change communication back to ASCII, you must write the value 1 into register **V30**. After writing to register V30, a power cycle must be performed before the setting takes effect. Note that the setting is automatically stored to flash when written.

See "Modbus RTU Addressing" for details on how to write the value.

3. Sample Usage

Achieving functionality via variables

Only the general purpose variables V1-V30 of the controller are accessible via Modbus-RTU communication. By reading/updating these variables, the desired functionality can be achieved by referencing them within a running stand-alone program. For details regarding stand-alone programming, see the user manual specific to your controller.

In a typical application, the controller runs a stand-alone program while the Modbus master accesses variables over the Modbus-RTU protocol.

The following is a sample of such a stand-alone program.

Standalone Example Program 1

```
EO=1                      ;* Enable the motor power

WHILE 1=1                 ;* Forever loop
      GOSUB 1             ;* Set speed settings
      IF V1=1
            X1000         ;* If V1 is set to 1 then go to position 1000
            WAITX         ;* Wait for move to complete
      ELSEIF V1=2
            X3000         ;* If V1 is set to 2 then go to position 3000
            WAITX         ;* Wait for move to complete
      ELSEIF V1=3
            X0            ;* If V1 is set to 3 then go to position 0
            WAITX         ;* Wait for move to complete
      ENDIF

      V2 = PX             ;* Store pulse position in V2

ENDWHILE            ;* Go back to WHILE statement

END

SUB 1
      HSPD=V4                 ;* Set the high speed to V4 pulses/sec
      LSPD=V5                 ;* Set the low speed to V5 pulses/sec
      ACC=V6                  ;* Set the acceleration to V6 msec
END SUB
```

4. Modbus RTU Addressing

The Arcus implementation of Modbus-RTU supports the holding register data type only. Coils, Discrete Inputs and Input Registers are not supported.

Data Model

Data Type Parameter	Size (Bits)	Data Type Description	Access	Address Range
03h	16	Holding Registers	Read/Write	[1-60]

Holding Registers – Read/Write – 16 bits

Address	Name	Description
1	V1_H	V1 Higher 16 bits [31:16]
2	V1_L	V1 Lower 16 bits [15:0]
3	V2_H	V2 Higher 16 bits [31:16]
4	V2_L	V2 Lower 16 bits [15:0]
5	V3_H	V3 Higher 16 bits [31:16]
6	V3_L	V3 Lower 16 bits [15:0]
7	V4_H	V4 Higher 16 bits [31:16]
8	V4_L	V4 Lower 16 bits [15:0]
9	V5_H	V5 Higher 16 bits [31:16]
10	V5_L	V5 Lower 16 bits [15:0]
11	V6_H	V6 Higher 16 bits [31:16]
12	V6_L	V6 Lower 16 bits [15:0]
13	V7_H	V7 Higher 16 bits [31:16]
14	V7_L	V7 Lower 16 bits [15:0]
15	V8_H	V8 Higher 16 bits [31:16]
16	V8_L	V8 Lower 16 bits [15:0]
17	V9_H	V9 Higher 16 bits [31:16]
18	V9_L	V9 Lower 16 bits [15:0]
19	V10_H	V10 Higher 16 bits [31:16]
20	V10_L	V10 Lower 16 bits [15:0]
21	V11_H	V11 Higher 16 bits [31:16]
22	V11_L	V11 Lower 16 bits [15:0]
23	V12_H	V12 Higher 16 bits [31:16]
24	V12_L	V12 Lower 16 bits [15:0]
25	V13_H	V13 Higher 16 bits [31:16]
26	V13_L	V13 Lower 16 bits [15:0]
27	V14_H	V14 Higher 16 bits [31:16]

28	V14_L	V14 Lower 16 bits [15:0]
29	V15_H	V15 Higher 16 bits [31:16]
30	V15_L	V15 Lower 16 bits [15:0]
31	V16_H	V16 Higher 16 bits [31:16]
32	V16_L	V16 Lower 16 bits [15:0]
33	V17_H	V17 Higher 16 bits [31:16]
34	V17_L	V17 Lower 16 bits [15:0]
35	V18_H	V18 Higher 16 bits [31:16]
36	V18_L	V18 Lower 16 bits [15:0]
37	V19_H	V19 Higher 16 bits [31:16]
38	V19_L	V19 Lower 16 bits [15:0]
39	V20_H	V20 Higher 16 bits [31:16]
40	V20_L	V20 Lower 16 bits [15:0]
41	V21_H	V21 Higher 16 bits [31:16]
42	V21_L	V21 Lower 16 bits [15:0]
43	V22_H	V22 Higher 16 bits [31:16]
44	V22_L	V22 Lower 16 bits [15:0]
45	V23_H	V23 Higher 16 bits [31:16]
46	V23_L	V23 Lower 16 bits [15:0]
47	V24_H	V24 Higher 16 bits [31:16]
48	V24_L	V24 Lower 16 bits [15:0]
49	V25_H	V25 Higher 16 bits [31:16]
50	V25_L	V25 Lower 16 bits [15:0]
51	V26_H	V26 Higher 16 bits [31:16]
52	V26_L	V26 Lower 16 bits [15:0]
53	V27_H	V27 Higher 16 bits [31:16]
54	V27_L	V27 Lower 16 bits [15:0]
55	V28_H	V28 Higher 16 bits [31:16]
56	V28_L	V28 Lower 16 bits [15:0]
57	V29_H	V29 Higher 16 bits [31:16]
58	V29_L	V29 Lower 16 bits [15:0]
†59	V30_H	V30 Higher 16 bits [31:16]
†60	V30_L	V30 Lower 16 bits [15:0]

†V30 is reserved for communication mode setting. In order to change from Modbus communication to ASCII, set V30=1.

Dealing with the Swapped-Long Registers

Variables V1-V30 are 32-bit twos-complement numbers. Since standard Modbus holding registers only hold 16 bits, the full 32-bit numbers are stored into two different registers. The lower addressed register represents the higher 16 bits of the 32-bit number, while higher addressed register represents the lower 16 bits of the 32-bit number. This is known as swapped-long.

Example 1: Set **V1** to: -34930493 (0xFDEB00C3)

       V1 holding register is found at address 1.

       Set Holding Register Address 1 [bits 16:31] to 0xFDEB
       Set Holding Register Address 2 [bits 0:15] to 0x00C3

Example 2: Read **V2** register

       V2 holding register is found

       Assuming that the values stored in these registers are:

       Holding Register Address 3 [bits 16:31] = 0x0000
       Holding Register Address 4 [bits 0:15] = 0x4933

       When reading this value, the full 32-bit value is: +18739 (0x00004933)
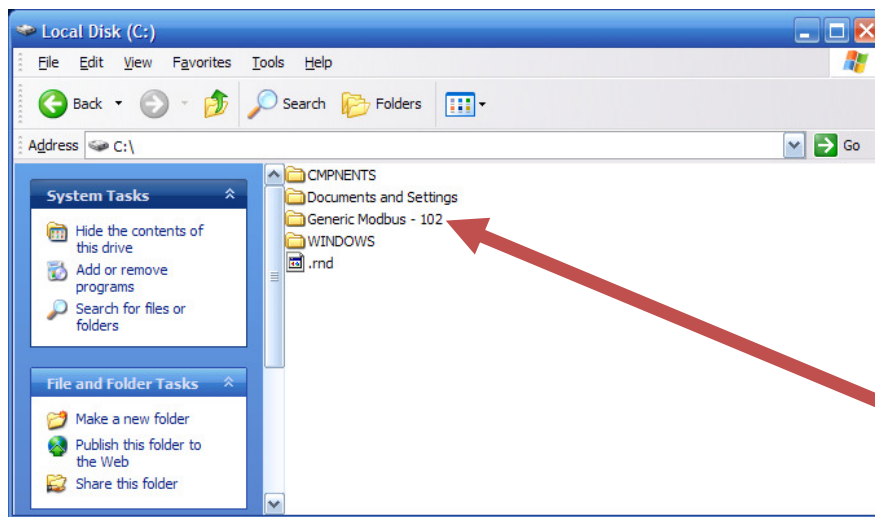
4. Modbus GUI

Arcus provides a GUI that allows Modbus communication to be achieved between a PC and the controller. Visit our website to download this software.
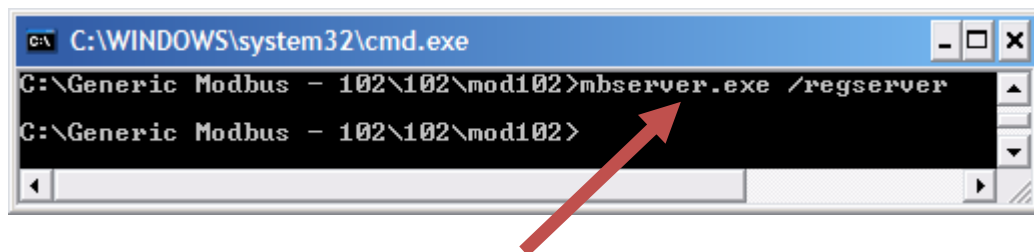
www.arcus-technology.com

Before running the program, make sure to register the 3rd party Modbus master driver on your PC. See instructions below:
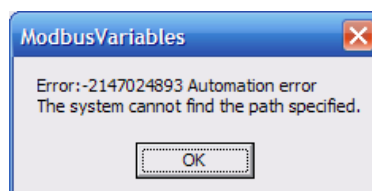
a. Download and extract GUI folder onto your PC.

b. Using the command line, browse to the "mod102" directory within the GUI folder and type "**mbserver.exe /regserver**" and press enter.

c. Note that if this process is not done correctly, you will receive the following error when trying to run the program.
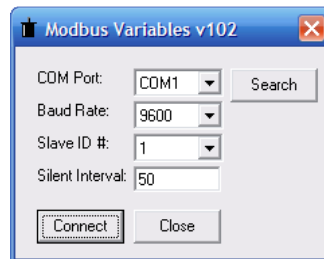
Connection Configuration

After the 3rd party Modbus driver has been successfully registered, start the program and the following screen should show up.
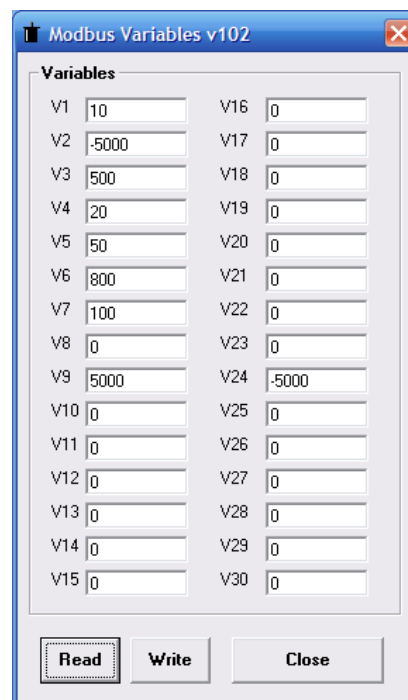
Select the correct COM port, Baud Rate and Slave ID (Modbus Device Number).

The search function allows the user to search for a device on a COM port at a given baud rate.

Main Screen

The main screen allows the user to read/write variables V1-V30.

Contact Information

Arcus Technology, Inc.

3061 Independence Drive. Suite H
Livermore, CA 94551
925-373-8800

www.arcus-technology.com