SmartFusion Modbus Reference Design

User’s Guide

Microsemi®

Table of Contents

Introduction

Overview

Modbus is a serial communications and application level protocol primarily targeting industrial network communications. The origins, rationale, and specifications for Modbus are summarized in the Modbus Organization FAQ [Reference 3], the Modbus Organization technical resources page [Reference 4], and also in various Modbus tutorials and introductory guides available on the web [Reference 9].

Reference Design Features

1. Modbus over serial line reference slave implementation based on a bare metal (non [Free]RTOS) implementation of the FreeModbus communications stack [Reference 6], targeting the Microsemi SmartFusion™ customizable system-on-chip (cSoC) device [Reference 2] A2F500-DEV-KIT board[1].
2. Supports Modbus serial line ASCII and RTU modes.
3. Supports RS-232 (point to point master and single slave) and RS-485 (bus-based multidrop master and multiple slaves) physical communication mediums.
4. Includes the complete Libero® System-on-Chip (SoC) software v10.0 SPB/MSS v2.4.105 hardware and SoftConsole (v3.3) firmware projects implementing the reference design Modbus slave which can be exercised using any third party Modbus master (including those which are PC hosted, mentioned in [Reference 9]) and which can be adapted and extended for customer specific requirements.
5. Supported Modbus functions—based on the FreeModbus communications stack, the reference design supports the following Modbus functions out of the box:

   a. Read Input Register (function code 0x04)

   b. Read Holding Registers (0x03)

   c. Write Single Register (0x06)

   d. Write Multiple Registers (0x10)

   e. Read/Write Multiple Registers (0x17)

   f. Read Coils (0x01)

   g. Write Single Coil (0x05)

   h. Write Multiple Coils (0x0F)

   i. Read Discrete Inputs (0x02)

   j. Report Slave ID (0x11)

   Refer to the FreeModbus API documentation [Reference 7] for information about extending the slave to support additional Modbus function codes.
6. Supports a variety of sample single bit read-only discrete input registers, single bit read-write coils registers, 16-bit read-only input registers and 16-bit read-write holding registers connected to board resources such as OLED (holding), LEDs (coils), DIP switches (discrete inputs), push-buttons (discrete inputs), and ACE analog voltage channel and RTC (inputs). The number and type of registers can be extended by the end user.

[1] *The reference design Libero SoC project can easily be retargeted to the SmartFusion A2F200-EVAL-KIT board, which supports only RS-232 (via the Silicon Laboratories CP2102 USB to UART bridge) communications and which does not support all of the hardware resources available on the A2F500-DEV-KIT board that are mapped to Modbus registers in the reference design slave implementation.*

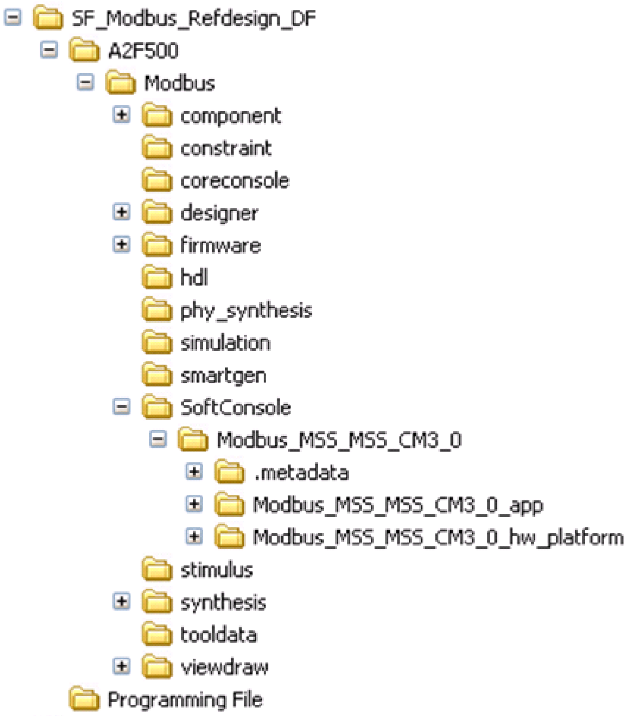
1 – Installing and Using the Reference Design

Installing the Reference Design

Download the design files from the Microsemi SoC Products Group website:

www.microsemi.com/soc/download/rsc/?f=MS_SF_Modbus_Ref_Design_DF

The reference design is delivered as a rar file and the contents should be extracted to a suitable folder on disk. Once extracted, the folder structure would be displayed as shown in Figure 1-1.



```
SF_Modbus_Refdesign_DF
    A2F500
        Modbus
            component
            constraint
            coreconsole
            designer
            firmware
            hdl
            phy_synthesis
            simulation
            smartgen
            SoftConsole
                Modbus_MSS_MSS_CM3_0
                    .metadata
                    Modbus_MSS_MSS_CM3_0_app
                    Modbus_MSS_MSS_CM3_0_hw_platform
            stimulus
            synthesis
            tooldata
            viewdraw
    Programming File
```

***Figure 1-1* • Folder Structure**

To get started with the reference design follow these steps:

1. Extract the contents of the reference design rar file.
2. Open the hardware project ..\SF_Modbus_Refdesign_DF\A2F500\Modbus\Modbus.prj in Libero SoC IDE v10.0.
3. If necessary, bring the design through the project flow to create an STP for programming the target A2F500-DEV-KIT board.
4. Program the resulting STP to the A2F500-DEV-KIT board using FlashPro from within Libero SoC.
5. Launch the SoftConsole project and double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window.
6. Build the Release target. Right-click the **Modbus_MSS_MSS_CM3_0_app** and **Modbus_MSS_MSS_CM3_0_hw_platform** projects in the SoftConsole Project Explorer and choose **Build Configurations** > **Set Active** > **Release** and then right-click again and choose **Build Project**.

7. Download the release build to SmartFusion MSS Embedded NVM (ENVM). Right-click the Modbus_MSS_MSS_CM3_0_app project in the SoftConsole Project Explorer and choose **DebugAs** > **Debug Configurations**. In the Debug Configurations dialog, select the Modbus_MSS_MSS_CM3_0_app Debug in ENVM debug configuration and choose **Debug**. The release build firmware should download to ENVM, start running, and then stop at the breakpoint at main().
8. Terminate the debug session and close SoftConsole.
9. Power cycle the A2F500-DEV-KIT board and the firmware should run, displaying the following message on the OLED, lighting LEDs D2 and D4.

```
Microsemi Corp v1.0
SmartFusion Modbus
```

Board Settings

The design example is made to be working on SmartFusion development kit board and SmartFusion evaluation kit board with default board settings. Refer the following user's guides for default board settings:

- SmartFusion Development Kit User's Guide
- SmartFusion Evaluation Kit User's Guide

Default Communication Settings

The default communication settings are as follows:

- Modbus serial RTU mode
- Modbus slave address 0x01
- MSS UART_0/RS-232 physical layer communications (via the SmartFusion cSoC board's Silicon Laboratories CP2102 USB to UART bridge)
- 19200 baud
- 8 data bits, as required by Modbus RTU mode. If the firmware is reconfigured to run in ASCII mode, the Modbus master must be configured to use 7 data bits, as required by Modbus ASCII.
- Even parity
- 1 stop bit

When connecting a Modbus master to the reference design slave, connect a USB cable from your PC to the A2F500-DEV-KIT board's MSS UART_0/Silicon Laboratories CP2102 USB to UART bridge mini USB connector and make sure that you know what port number is allocated to the virtual USB COM port before configuring the master Modbus and communication settings. If you reconfigure the firmware to use MSS UART_1/RS-485, refer to the "RS-485 Communications" section on page 27.

Using Modpoll

Modpoll® is a simple command line read-only freeware Modbus master available from proconX Pty Ltd. [Reference 9]. Download and install/extract Modpoll, open a command shell, and change directory to the folder containing the modpoll.exe executable. Modpoll.exe -h displays help about the different command line options supported.

Read Input Registers

To query the reference design slave's two 16-bit read-only input registers, which store the RTC in seconds and RV1 3.3 V pot voltage in mV, run Modpoll as follows:

```
modpoll.exe -m rtu -a 1 -r 1 -c 2 -t 3 -b 19200 -d 8 -p even COM4
```

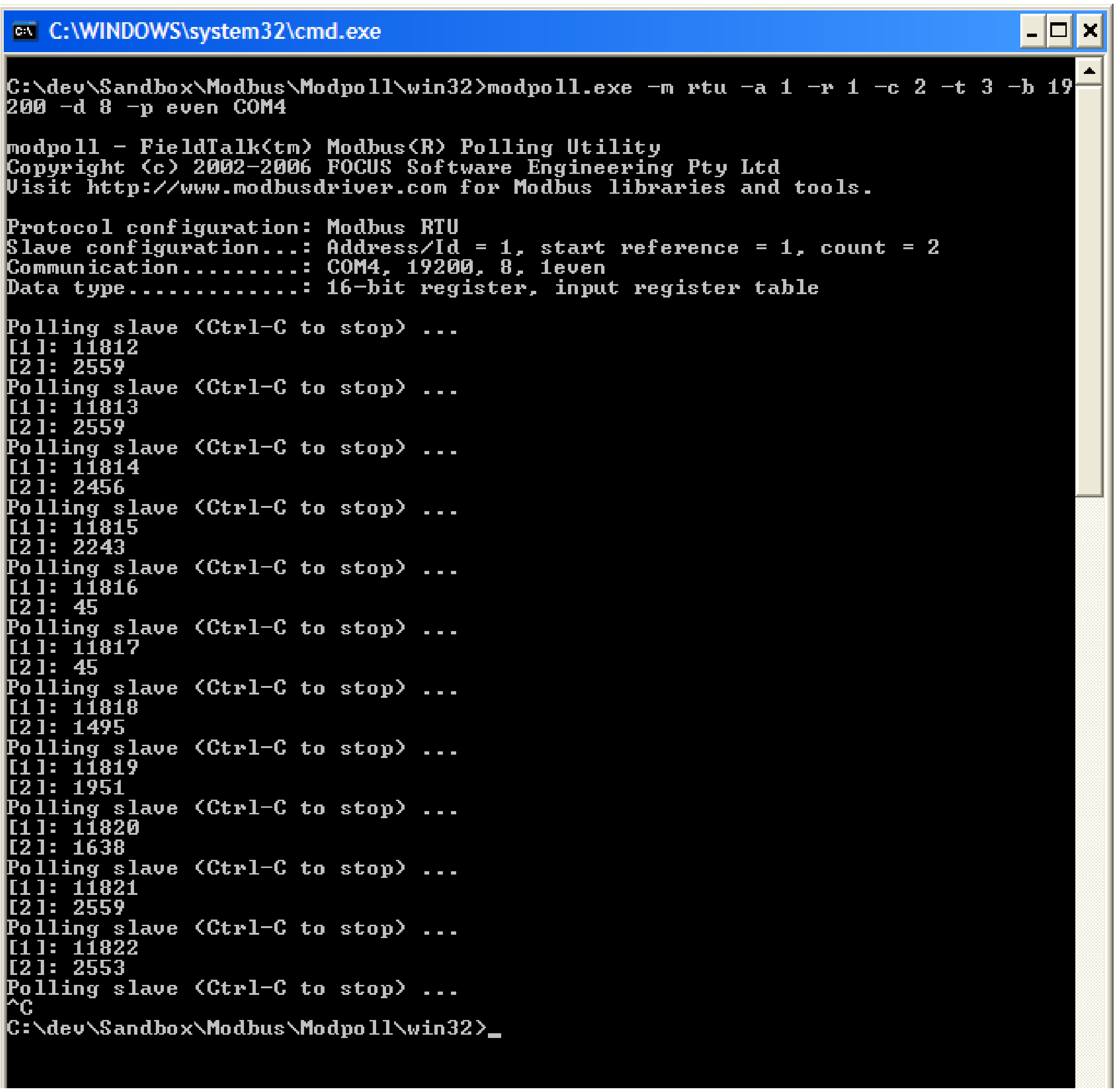
The command line parameters are shown in Table 1-1:

***Table 1-1* • Description of Command Line Parameters**

Command Line Option	Description
-m rtu	Modbus serial mode.
-a 1	Modbus target slave address.
-r 1	Offset from start of the relevant Modbus register block (as determined by the -t command line option below) to start reading.
-c 2	Number of values to poll.
-t 3	Poll 16-bit read-only input registers. See Table 2-3 on page 23 for details of the reference design slave's Modbus register map.
-b	19200 Baud rate.
-p even	Even parity.
COM4	The PC virtual COM port for connecting to the A2F500-DEV-KIT board. Modify this according to your local setup.

Modpoll will continuously poll the two input registers supported by the reference design slave (RTC and RV1 pot voltage).

You should see the first register (RTC) counting up in seconds and the second one varying as you turn the RV1 pot on the A2F500-DEV-KIT board (Figure 1-2).



```
C:\dev\Sandbox\Modbus\Modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 2 -t 3 -b 19
200 -d 8 -p even COM4

modpoll - FieldTalk(tm) Modbus(R) Polling Utility
Copyright (c) 2002-2006 FOCUS Software Engineering Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: Address/Id = 1, start reference = 1, count = 2
Communication.........: COM4, 19200, 8, 1even
Data type.............: 16-bit register, input register table

Polling slave (Ctrl-C to stop) ...
[1]: 11812
[2]: 2559
Polling slave (Ctrl-C to stop) ...
[1]: 11813
[2]: 2559
Polling slave (Ctrl-C to stop) ...
[1]: 11814
[2]: 2456
Polling slave (Ctrl-C to stop) ...
[1]: 11815
[2]: 2243
Polling slave (Ctrl-C to stop) ...
[1]: 11816
[2]: 45
Polling slave (Ctrl-C to stop) ...
[1]: 11817
[2]: 45
Polling slave (Ctrl-C to stop) ...
[1]: 11818
[2]: 1495
Polling slave (Ctrl-C to stop) ...
[1]: 11819
[2]: 1951
Polling slave (Ctrl-C to stop) ...
[1]: 11820
[2]: 1638
Polling slave (Ctrl-C to stop) ...
[1]: 11821
[2]: 2559
Polling slave (Ctrl-C to stop) ...
[1]: 11822
[2]: 2553
Polling slave (Ctrl-C to stop) ...
^C
C:\dev\Sandbox\Modbus\Modpoll\win32>_
```

Figure 1-2 • **Continuous Polling**

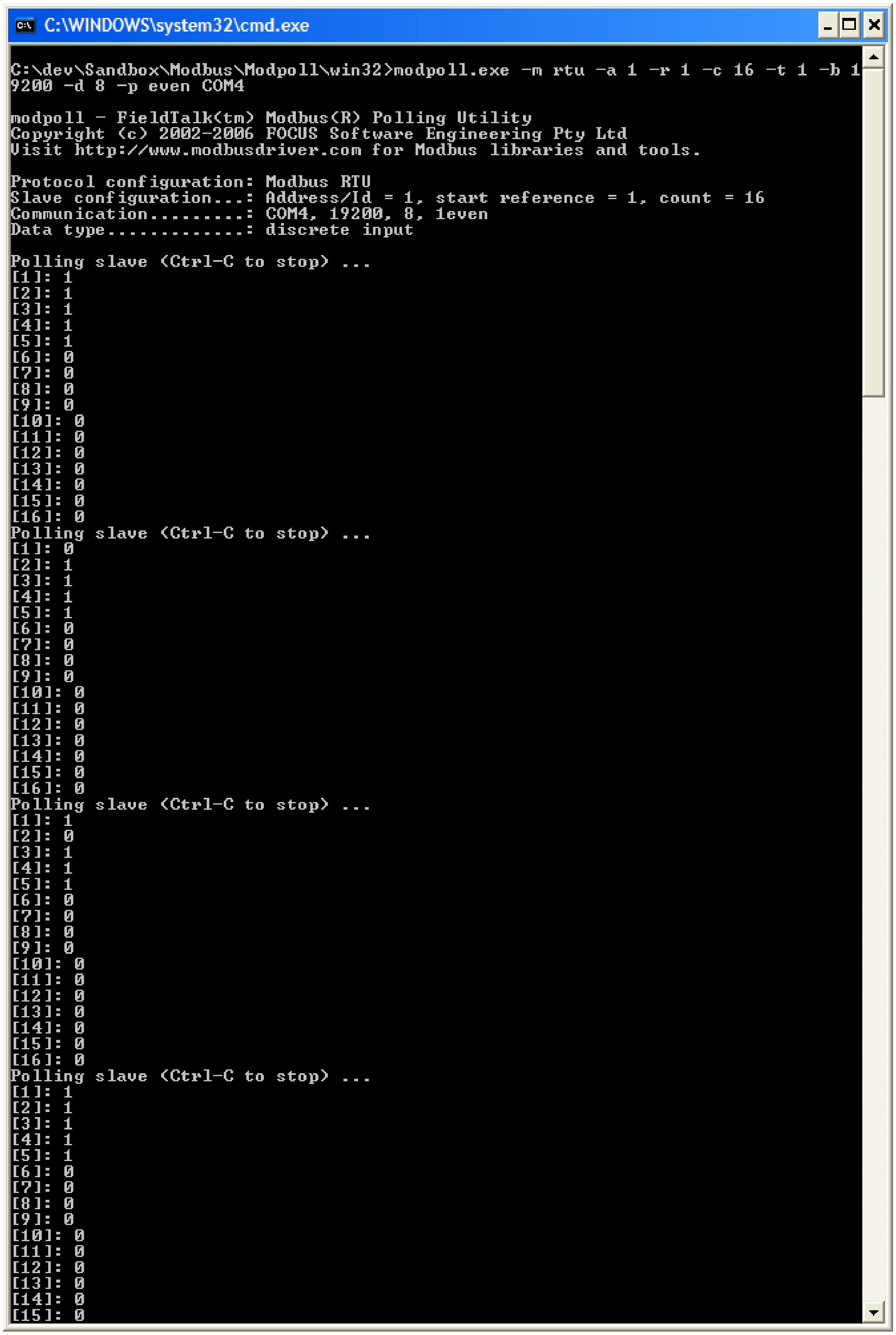
Read Discrete Input Registers

To query the 16 single-bit read-only discrete input registers, run Modpoll again with the following command line options:

```
modpoll.exe -m rtu -a 1 -r 1 -c 16 -t 1 -b 19200 -d 8 -p even COM4
```

This time use **-t 1 -c 16** to read the 16 single-bit read-only discrete inputs supported by the reference design slave. See Table 2-3 on page 23 for more information about the Modbus register map supported by the reference design slave.

As Modpoll polls the slave, toggle the A2F500-DEV-KIT board's SW1-SW5 push-buttons and A2F_DIP DIP switches to see the effect that this has on the results reported by Modpoll (Figure 1-3).



```
C:\dev\Sandbox\Modbus\Modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 16 -t 1 -b 1
9200 -d 8 -p even COM4

modpoll - FieldTalk(tm) Modbus(R) Polling Utility
Copyright (c) 2002-2006 FOCUS Software Engineering Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: Address/Id = 1, start reference = 1, count = 16
Communication.........: COM4, 19200, 8, 1even
Data type.............: discrete input

Polling slave (Ctrl-C to stop) ...
[1]: 1
[2]: 1
[3]: 1
[4]: 1
[5]: 1
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
[11]: 0
[12]: 0
[13]: 0
[14]: 0
[15]: 0
[16]: 0
Polling slave (Ctrl-C to stop) ...
[1]: 0
[2]: 1
[3]: 1
[4]: 1
[5]: 1
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
[11]: 0
[12]: 0
[13]: 0
[14]: 0
[15]: 0
[16]: 0
Polling slave (Ctrl-C to stop) ...
[1]: 1
[2]: 0
[3]: 1
[4]: 1
[5]: 1
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
[11]: 0
[12]: 0
[13]: 0
[14]: 0
[15]: 0
[16]: 0
Polling slave (Ctrl-C to stop) ...
[1]: 1
[2]: 1
[3]: 1
[4]: 1
[5]: 1
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
[11]: 0
[12]: 0
[13]: 0
[14]: 0
[15]: 0
```

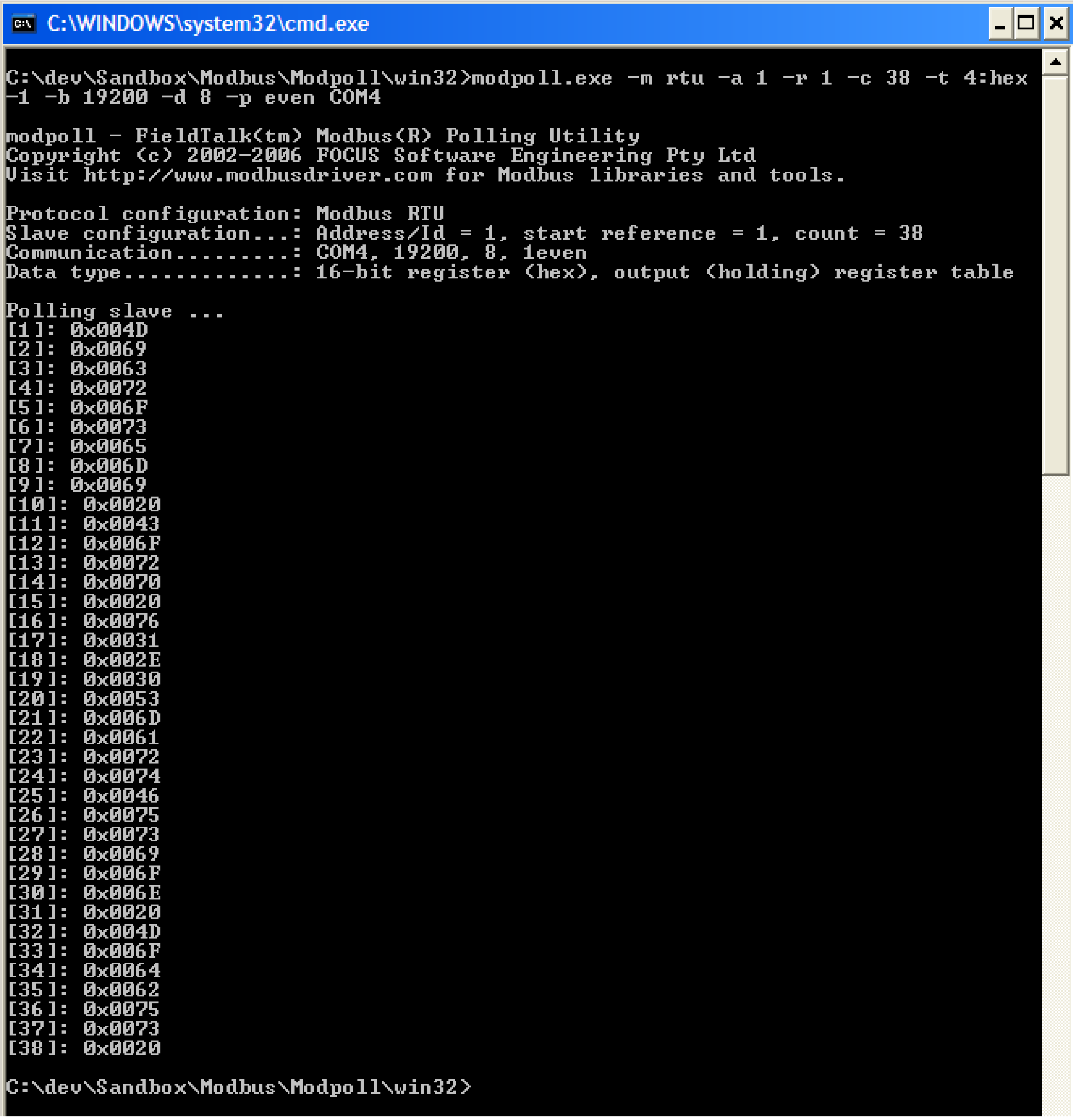
***Figure 1-3* • Query Discrete Input Registers**

Read Holding Registers

The 38 16-bit read-write holding registers contain (in their lower 8 bits) the values used to display characters in each of the OLED's 38 (2 row x 19 column) character positions. Modpoll is a read-only Modbus master, so can read but not write these. To read these registers, run Modpoll as follows:

```
modpoll.exe -m rtu -a 1 -r 1 -c 38 -t 4:hex -1 -b 19200 -d 8 -p even COM4
```

In this case, **-c 38** tells Modpoll to read 38 registers, -t 4:hex tells it to read the holding registers and display them in hex, and -1 tells it to poll once rather than continuously (Figure 1-4).



```
C:\dev\Sandbox\Modbus\Modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 38 -t 4:hex
-1 -b 19200 -d 8 -p even COM4

modpoll - FieldTalk(tm) Modbus(R) Polling Utility
Copyright (c) 2002-2006 FOCUS Software Engineering Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: Address/Id = 1, start reference = 1, count = 38
Communication.........: COM4, 19200, 8, 1even
Data type.............: 16-bit register (hex), output (holding) register table

Polling slave ...
[1]: 0x004D
[2]: 0x0069
[3]: 0x0063
[4]: 0x0072
[5]: 0x006F
[6]: 0x0073
[7]: 0x0065
[8]: 0x006D
[9]: 0x0069
[10]: 0x0020
[11]: 0x0043
[12]: 0x006F
[13]: 0x0072
[14]: 0x0070
[15]: 0x0020
[16]: 0x0076
[17]: 0x0031
[18]: 0x002E
[19]: 0x0030
[20]: 0x0053
[21]: 0x006D
[22]: 0x0061
[23]: 0x0072
[24]: 0x0074
[25]: 0x0046
[26]: 0x0075
[27]: 0x0073
[28]: 0x0069
[29]: 0x006F
[30]: 0x006E
[31]: 0x0020
[32]: 0x004D
[33]: 0x006F
[34]: 0x0064
[35]: 0x0062
[36]: 0x0075
[37]: 0x0073
[38]: 0x0020

C:\dev\Sandbox\Modbus\Modpoll\win32>
```

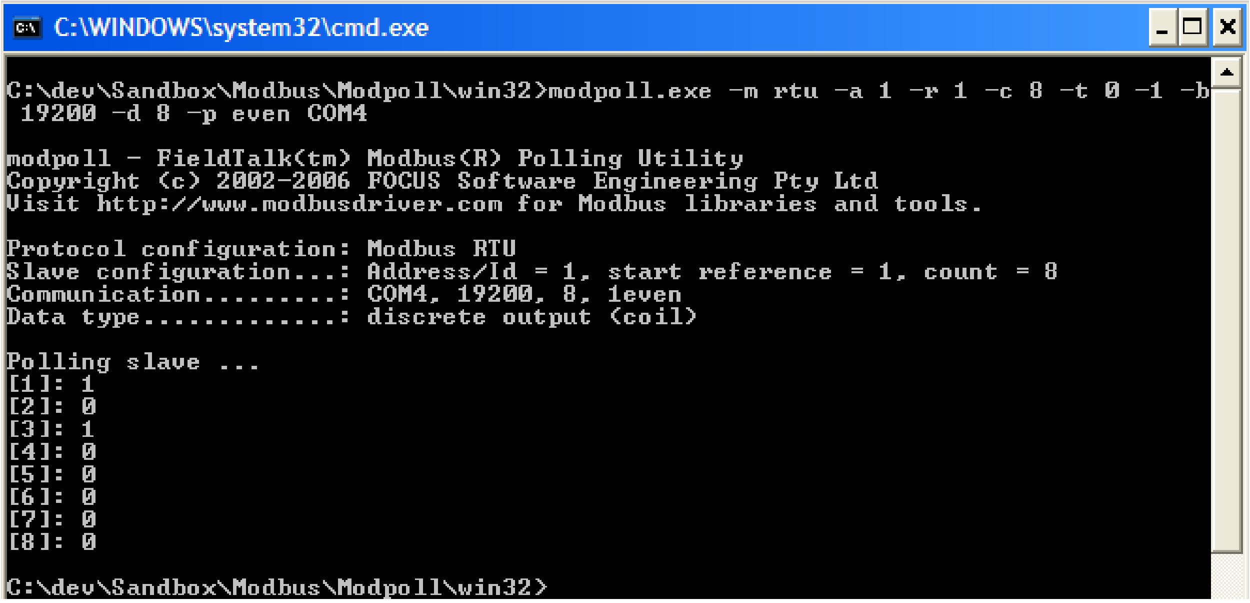
Figure 1-4 **• Hex Display**

Read Coils Registers

To read eight 1-bit read-write coils registers (four of which are connected to the A2F500-DEV-KIT board's LEDs), run Modpoll as follows:

```
modpoll.exe -m rtu -a 1 -r 1 -c 8 -t 0 -1 -b 19200 -d 8 -p even COM4
```

Note that the LEDs are active low, so a 0 means that the LED is on while 1 means that it is off (Figure 1-5).



```
C:\dev\Sandbox\Modbus\Modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 8 -t 0 -1 -b
 19200 -d 8 -p even COM4

modpoll - FieldTalk(tm) Modbus(R) Polling Utility
Copyright (c) 2002-2006 FOCUS Software Engineering Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: Address/Id = 1, start reference = 1, count = 8
Communication.........: COM4, 19200, 8, 1even
Data type.............: discrete output (coil)

Polling slave ...
[1]: 1
[2]: 0
[3]: 1
[4]: 0
[5]: 0
[6]: 0
[7]: 0
[8]: 0

C:\dev\Sandbox\Modbus\Modpoll\win32>
```

Figure 1-5 • **Read Coils Registers**

Using Automated Solution's MiniHMI

Automated Solutions Inc. [Reference 9] provides commercial software solutions for HMI and SCADA developers, including various Modbus solutions. Their product range includes a Modbus RTU/ASCII Master ActiveX Control and some example applications. Contact Automated Solutions Inc. or refer to their website for details about their commercial tools offerings and prices.

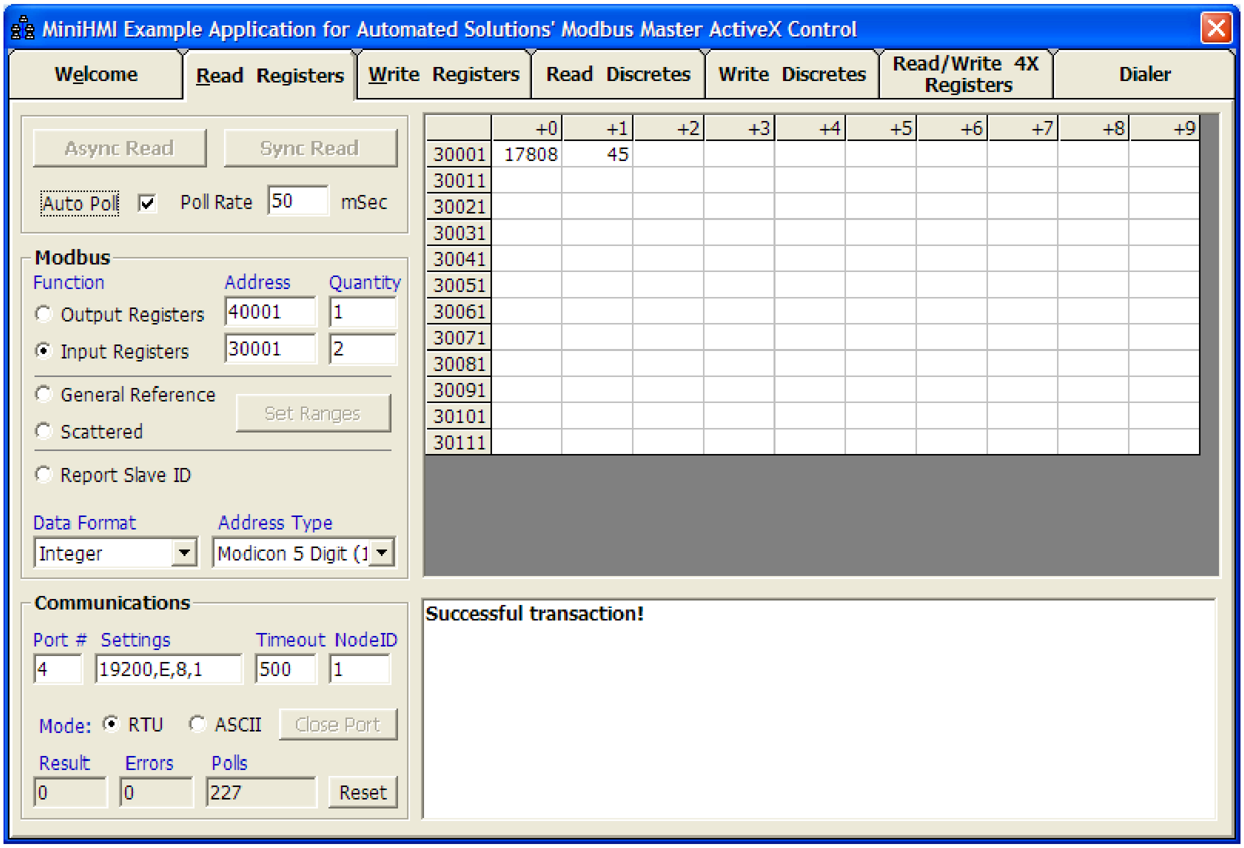
Automated Solutions Inc. also provides a free, full featured 30-day trial version of their Modbus RTU/ASCII Master ActiveX Control and example applications which can be used to exercise and demonstrate the features of the Microsemi SmartFusion Modbus reference design slave.

Obtain and install the demo or full version of the Automated Solutions Inc. Modbus RTU/ASCII Master ActiveX Control package. Refer to the documentation and help provided with the package and on the Automated Solutions Inc. website for more details about the capabilities of the ActiveX component and example applications.

The MiniHMI example application can be run from **Start** > **All Programs** > **Automated Solutions ActiveX** > **Modbus Master** > **MiniHMI Example Application**.

Read Input Registers

Run the MiniHMI example application. Click the **Read Registers** tab. Ensure that the Communications settings are configured appropriately to match your slave setup. See the "Default Communication Settings" section on page 8 for the default settings. The MiniHMI settings with the possible exception of Communications Port match the reference design slave default settings. Select the **Modbus** > **Function** > I**nput Registers** radio button and in the Quantity field enter **2**. Check the **Auto Poll** check box and MiniHMI should start, continuously polling the RTC and RV1 3.3 V pot input registers. The values of the registers should update as time passes and you rotate the RV1 pot. Clear the Auto Poll check box to stop continuous polling.



Auto Poll ☑ Poll Rate 50 mSec

Modbus

Function	Address	Quantity
○ Output Registers	40001	1
● Input Registers	30001	2
○ General Reference		
○ Scattered		
○ Report Slave ID		

Data Format: Integer
Address Type: Modicon 5 Digit (1

Communications

Port #	Settings	Timeout	NodeID
4	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII

Result	Errors	Polls
0	0	227

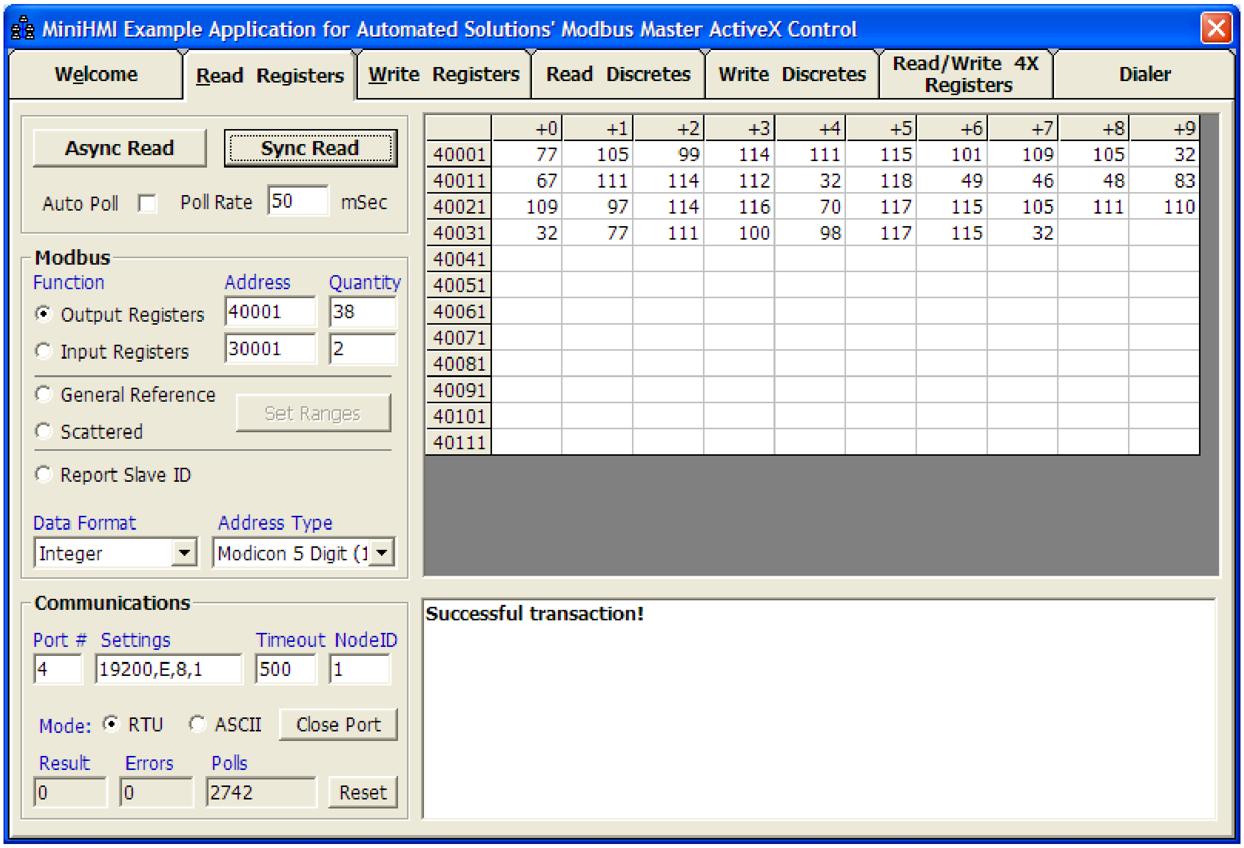
	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
30001	17808	45								
30011										
30021										
30031										
30041										
30051										
30061										
30071										
30081										
30091										
30101										
30111										

Successful transaction!

***Figure 1-6* • Polling with MiniHMI**

Read/Write Holding Registers

On the Read Registers tab, select the **Modbus** > **Function** > **Output Registers** radio button. Enter **38** in the Quantity field. Click the **Sync Read** button and MiniHMI should read back the 38 OLED character position holding registers.



Auto Poll □ Poll Rate 50 mSec

Modbus

Function	Address	Quantity
● Output Registers	40001	38
○ Input Registers	30001	2

○ General Reference
○ Scattered
○ Report Slave ID

Data Format: Integer
Address Type: Modicon 5 Digit (1

Communications

Port #	Settings	Timeout	NodeID
4	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII

Result	Errors	Polls
0	0	2742

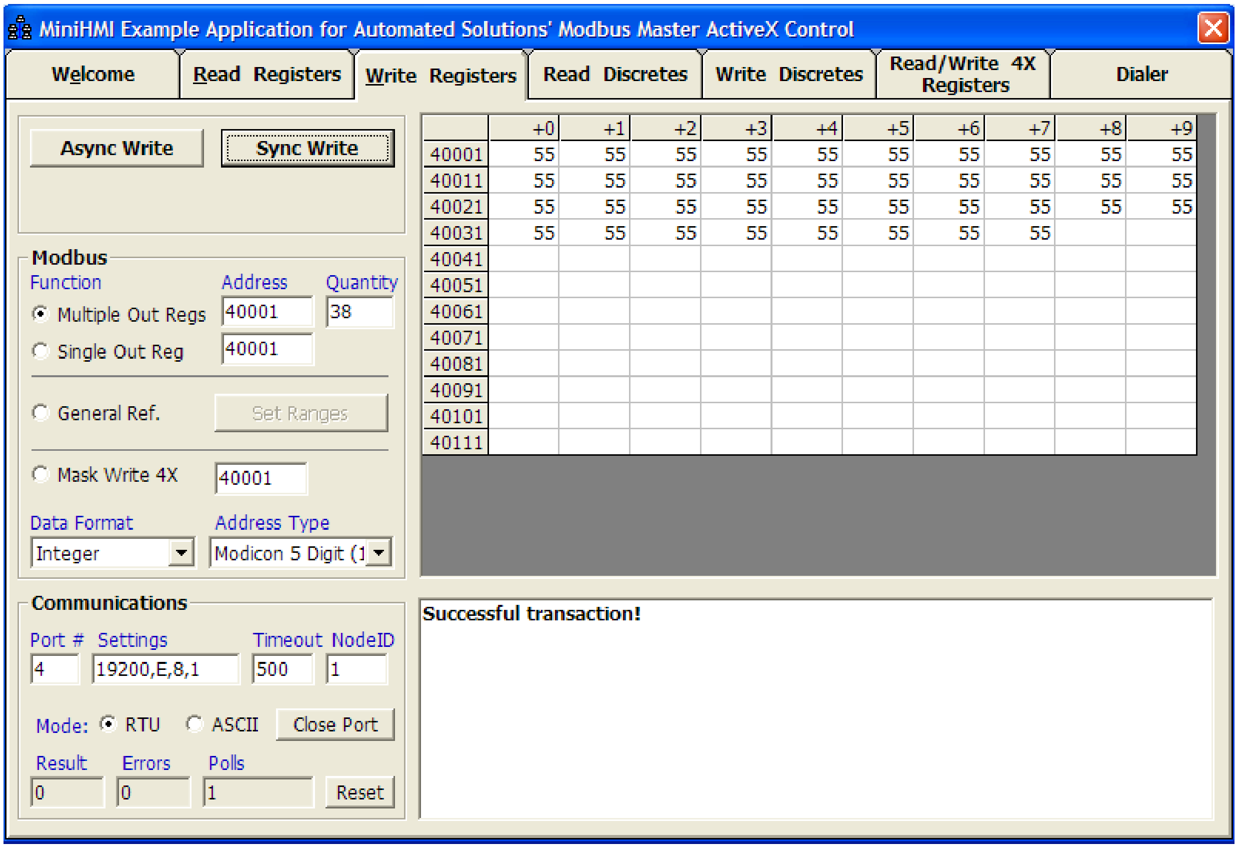
	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
40001	77	105	99	114	111	115	101	109	105	32
40011	67	111	114	112	32	118	49	46	48	83
40021	109	97	114	116	70	117	115	105	111	110
40031	32	77	111	100	98	117	115	32		
40041										
40051										
40061										
40071										
40081										
40091										
40101										
40111										

Successful transaction!

Figure 1-7 • **Read Holding Registers**

To write to the holding registers in order to change what is displayed on the OLED, go to the **Write Registers** tab. Ensure that the Communications settings are configured correctly ("Default Communication Settings" section on page 8). Note that the Communications settings must be set independently on each tab and are not retained between runs.

Select the **Modbus** > **Function** > **Multiple Out Regs** radio button and enter a value between 1 and 38 in the Quantity field. In the register grid/spreadsheet view, enter as many values as you want to write to the OLED holding registers. All 16 bits of the values will be stored but only the lower 8 bits will be written to the OLED. Click the **Sync Write** button to flush the new values to the reference design Modbus slave. For example, to write 7’s to all OLED character positions, do the following:



Modbus

Function	Address	Quantity
● Multiple Out Regs	40001	38
○ Single Out Reg	40001	

○ General Ref. Set Ranges

○ Mask Write 4X 40001

Data Format: Integer

Address Type: Modicon 5 Digit (1

Communications

Port #	Settings	Timeout	NodeID
4	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII Close Port

Result	Errors	Polls
0	0	1

Reset

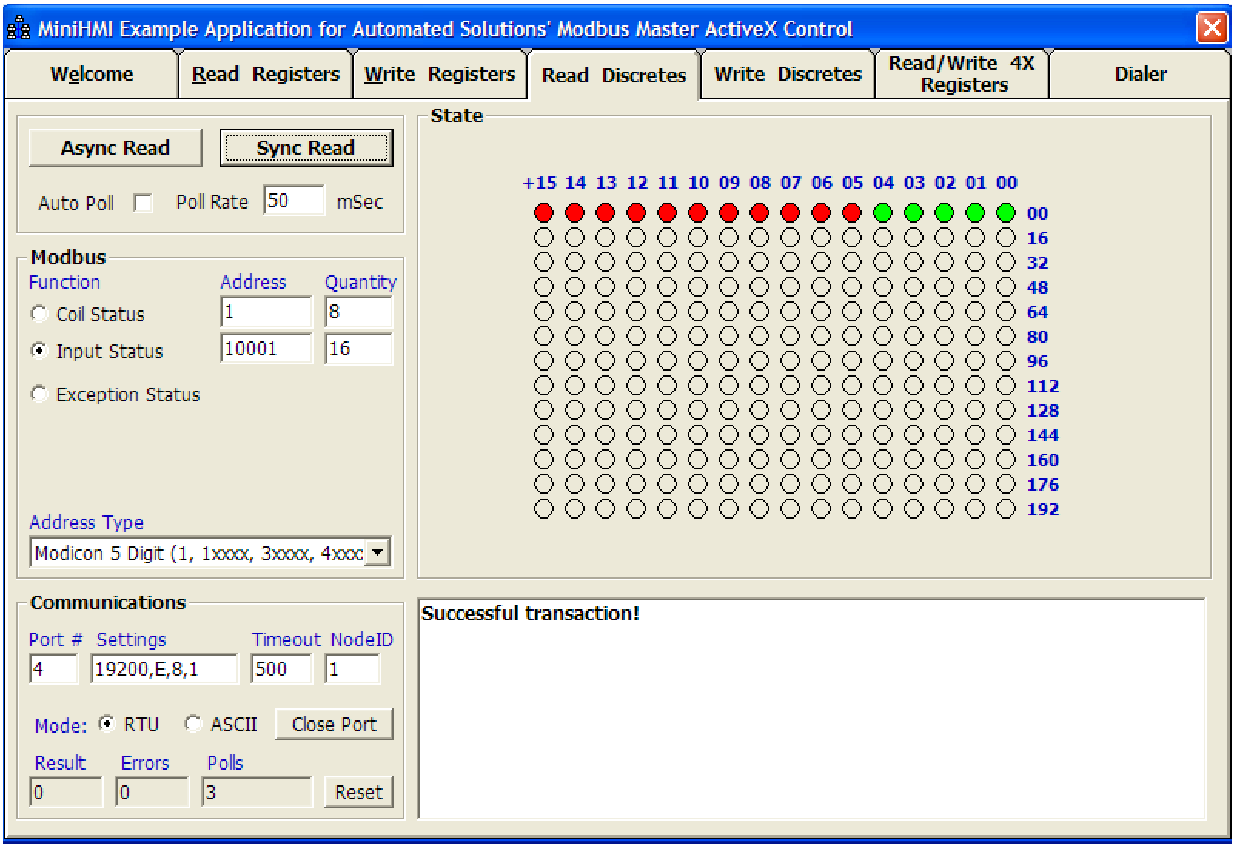
	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
40001	55	55	55	55	55	55	55	55	55	55
40011	55	55	55	55	55	55	55	55	55	55
40021	55	55	55	55	55	55	55	55	55	55
40031	55	55	55	55	55	55	55	55		
40041										
40051										
40061										
40071										
40081										
40091										
40101										
40111										

Successful transaction!

Figure 1-8 • **Write to OLED Registers**

Read Discrete Inputs and Coils

To read the discrete inputs and coils, go to the **Read Discretes** tab. Ensure the Communications settings are set correctly, choose to read 8 or fewer coils or 16 or fewer discrete inputs and then click **Sync Read** to read once or Auto Poll to poll continuously. When reading the 16 discrete inputs, the MiniHMI GUI's State view will reflect changes due to manual toggling of the A2F500-DEV-KIT board's SW1-5 push-buttons or A2F_DIPS DIP switches.



MiniHMI Example Application for Automated Solutions' Modbus Master ActiveX Control

Auto Poll □ Poll Rate 50 mSec

Modbus

Function	Address	Quantity
○ Coil Status	1	8
● Input Status	10001	16
○ Exception Status		

Address Type: Modicon 5 Digit (1, 1xxxx, 3xxxx, 4xxx

Communications

Port #	Settings	Timeout	NodeID
4	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII

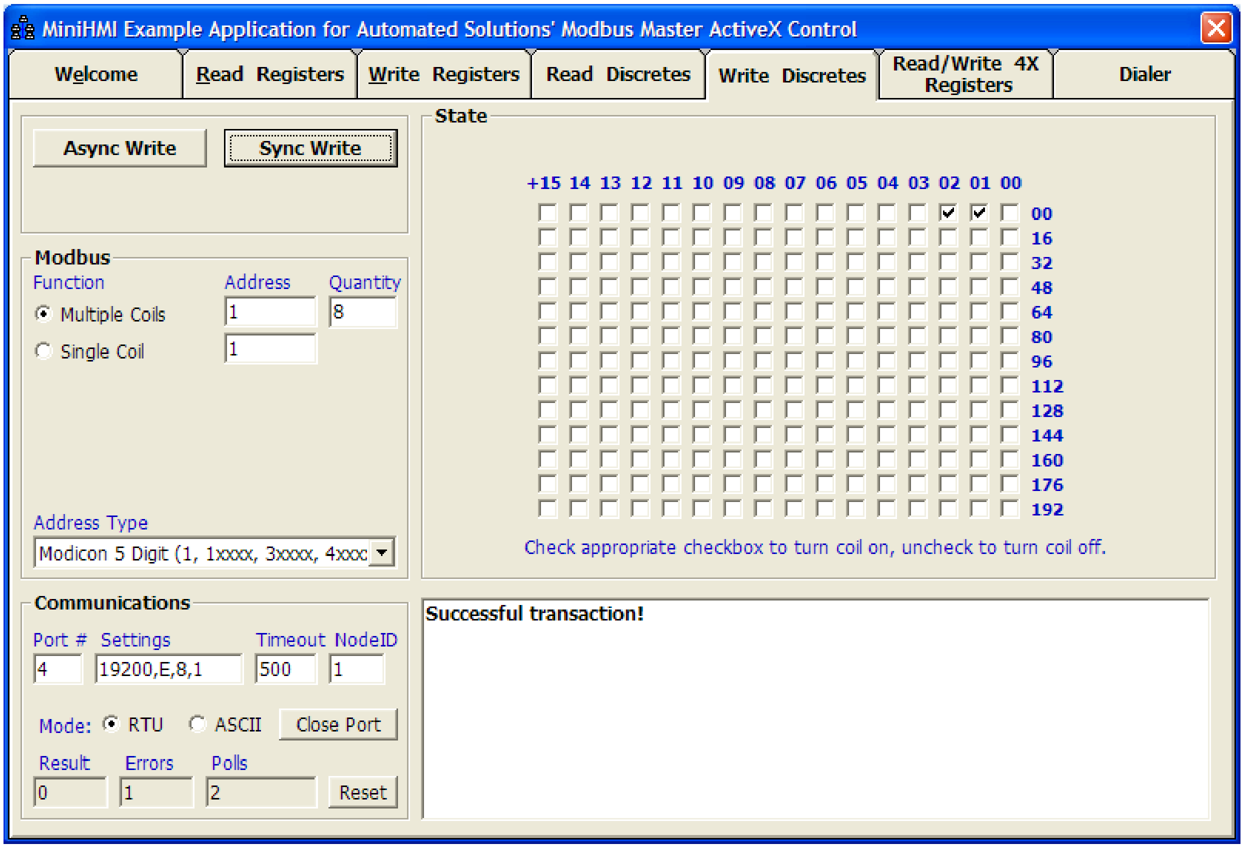
Result	Errors	Polls
0	0	3

Successful transaction!

Figure 1-9 • **Read Discrete Inputs and Coils**

Write Coils

To write the coils (LEDs), go to the **Write Discretes** tab. Ensure that the Communications settings are configured appropriately. Select **Single Coils** or **Multiple Coils** and enter a Quantity between 1 and 4 (or up to 8 but only coils 1-4 are actually connected to board resources/LEDs by default). Toggle any of the first four coils (00-04) in the State view and then click the **Sync Write** button to change the status of the LEDs. You should see the changes reflected on the A2F-DEV-KIT board itself.



State

+15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☑	☐	00
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	16
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	32
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	48
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	64
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	80
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	96
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	112
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	128
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	144
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	160
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	176
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	192

Check appropriate checkbox to turn coil on, uncheck to turn coil off.

Modbus

Function	Address	Quantity
● Multiple Coils	1	8
○ Single Coil	1	

Address Type: Modicon 5 Digit (1, 1xxxx, 3xxxx, 4xxxx

Communications

Port #	Settings	Timeout	NodeID
4	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII

Result	Errors	Polls
0	1	2

Successful transaction!

Figure 1-10 • **Write Coils**

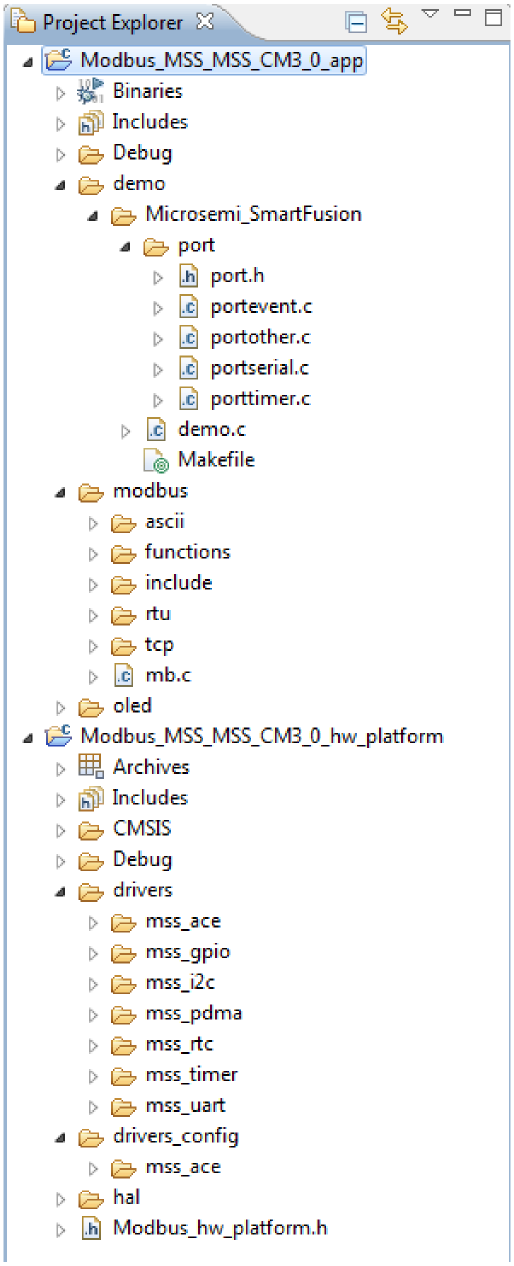
Using Other Modbus Masters

Similar to the way proconX Pty Ltd Modpoll and the Automated Solutions Inc. Modbus RTU/ASCII ActiveX Component MiniHMI example application can be used to interact with the Microsemi SmartFusion Reference Design sample Modbus slave, any other Modbus compatible PC hosted or other master can also be used. The main issue is that the master Modbus and serial communications settings match the slave target settings.

2 – SoftConsole Firmware Project

Project Layout

The screenshot of the SoftConsole v3.3 Project Explorer in Figure 2-1 outlines the structure of the firmware project. To invoke SoftConsole project, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window.



```
Project Explorer
Modbus_MSS_MSS_CM3_0_app
    Binaries
    Includes
    Debug
    demo
        Microsemi_SmartFusion
            port
                port.h
                portevent.c
                portother.c
                portserial.c
                porttimer.c
            demo.c
            Makefile
    modbus
        ascii
        functions
        include
        rtu
        tcp
        mb.c
    oled
Modbus_MSS_MSS_CM3_0_hw_platform
    Archives
    Includes
    CMSIS
    Debug
    drivers
        mss_ace
        mss_gpio
        mss_i2c
        mss_pdma
        mss_rtc
        mss_timer
        mss_uart
    drivers_config
        mss_ace
    hal
    Modbus_hw_platform.h
```

***Figure 2-1* • SoftConsole Project Explorer**

Table 2-1 summarizes the contents and purpose of each folder:

***Table 2-1* • Folder Description**

Folder/File	Description
Includes	Project level include files/folders
CMSIS	SmartFusion CMSIS-PAL
demo/Microsemi_SmartFusion/port	FreeModbus porting layer files targeting SmartFusion cSoC device
demo/Microsemi_SmartFusion/demo.c	Reference design slave implementation program file containing main(), default slave firmware configuration #defines, Modbus register and callback implementations etc.
drivers	Various SmartFusion firmware drivers
drivers/mss_ace	SmartFusion MSS ACE Driver
drivers/mss_gpio	SmartFusion MSS GPIO Driver
drivers/mss_i2c	SmartFusion MSS I2C Driver
drivers/mss_rtc	SmartFusion MSS RTC Driver
drivers/mss_timer	SmartFusion MSS Timer Driver
drivers/mss_uart	SmartFusion MSS UART Driver
drivers_config/mss_ace	MSS ACE driver configuration code generated from the SmartFusion MSS configurator based on the MSS ACE configuration
modbus	The core FreeModbus communication stack
oled	MSS I^{2}C based OLED driver for SmartFusion development/eval boards

Slave Firmware Configuration #defines

The following manifest constants control various aspects of the firmware operation and can be modified by specifying new values in the SoftConsole project properties (**Properties** > **C/C++ Build** > **Settings** > **GNU C Compiler** > **Symbols**) or by editing the #defines directly before recompiling the firmware, in Modbus_MSS_MSS_CM3_0_app/Microsemi_SmartFusion/demo.c.

Table 2-2 • **Manifest Constants**

Manifest Constant	**Description – Default Value**
MODBUS_SERIAL_MODE	FreeModbus communication stack mode of serial operation • MB_RTU– RTU mode • MB_ASCII – ASCII mode • Default: MB_RTU Note that in RTU/ASCII mode the Modbus master serial communications must be configured for 8/7 data bits respectively
MODBUS_SLAVE_ADDR	Modbus slave address. • 1 – 247 (0x01 – 0xF7) • Default: 1 (0x01)
MODBUS_PORT	Serial port used • 0 = MSS UART_0/RS-232 • 1 = MSS UART_1/RS-485* • Default: 0
MODBUS_BAUD_RATE	Baud rate • Default: 19200
MODBUS_PARITY	Parity • MB_PAR_EVEN • MB_PAR_ODD • MB_PAR_NONE • Default: MB_PAR_EVEN
MODBUS_SLAVEID	Modbus slave id; one byte ID followed by an optional number of bytes of device specific data • Default: 0x55 0xC0 0xFF 0xEE
REG_DISCRETE_START	Offset (from Modbus register address 10000) of first discrete input register implemented • Default: 1
REG_DISCRETE_NREGS	Number of discrete input registers implemented • Default: 2
REG_COILS_START	Offset (from Modbus register address 0) of first coil register implemented • Default: 1
REG_COILS_NREGS	Number of discrete input registers implemented • Default: 1
REG_INPUT_START	Offset (from Modbus register address 30000) of first input register implemented • Default: 1

*Note: *A2F500-DEV-KIT only – not supported on A2F200-EVAL-KIT*

Table 2-2 **• Manifest Constants (continued)**

Manifest Constant	Description – Default Value
REG_INPUT_NREGS	Number of input registers implemented • Default: 2
REG_HOLDING_START	Offset (from Modbus register address 40000) of first holding register implemented • Default: 1
REG_HOLDING_NREGS	Number of holding registers implemented • Default: 38

*Note: *A2F500-DEV-KIT only – not supported on A2F200-EVAL-KIT*

Default Configuration

By default, the reference design SoftConsole firmware project is configured to use the following Modbus and serial settings:

- Modbus serial RTU mode
- Modbus slave address 0x01
- Modbus slave id 0x55 with three optional data bytes 0xC0 0xFF 0xEE
- MSS UART_0/RS-232 physical layer communications (via the SmartFusion board's Silicon Laboratories CP2102 USB to UART bridge)
- 19200 baud
- 8 data bits (as required by Modbus RTU mode; Modbus ASCII mode uses 7 data bits)
- Even parity
- 1 stop bit

Linker Scripts

By default, the project is set up to use the following CMSIS-PAL sample linker scripts:

- Debug target: Modbus_MSS_MSS_CM3_0_hw_platform/CMSIS/startup_gcc/debug-in-actel-smartfusionesram.ld
- Release target: Modbus_MSS_MSS_CM3_0_hw_platform/CMSIS/startup_gcc/debug-in-actel-smartfusionenvm.ld

The debug target supports downloading to and debugging from SmartFusion MSS Embedded SRAM (ESRAM). The release target supports downloading to and debugging from SmartFusion MSS Embedded NVM (ENVM). Once the release target has been downloaded, it will persist in ENVM and will run from the board reset/power cycle. If you prefer to download the firmware via FlashPro, the release target can be modified to use the CMSIS CMSIS/startup_gcc/production-execute-in-place.ld linker script and the Intel HEX file resulting from the build process can be imported into an MSS ENVM Data Storage Client configured to load at offset 0x00000000 of ENVM so that the firmware runs from ENVM out of reset.

As with any SmartFusion firmware project, other build/link and memory configurations are possible, such as booting from ENVM, copying/relocating to ESRAM or external RAM and continuing to run from there, or more sophisticated "scatter loading" of portions of the firmware image to disparate memory regions. These configurations are beyond the scope of this document.

FreeModbus Configuration Options

See the FreeModbus API documentation [Reference 7] for information about manifest constants (in SmartFusion_demo/modbus/include/mbconfig.h) that control the configuration of the FreeModbus communications stack itself.

Reference Design Slave Modbus Register Map

The reference design slave firmware supports the Modbus registers shown in Table 2-3[1].

***Table 2-3* • Supported Modbus Registers**

Modbus Address	Physical Resource
Coils Registers – Single Bit Read/Write	
1	LED D1
2	LED D2
3	LED D3
4	LED D4
5	Not connected – reads as 0, writes ignored
6	Not connected – reads as 0, writes ignored
7	Not connected – reads as 0, writes ignored
8	Not connected – reads as 0, writes ignored
Discrete Input Registers – Single Bit Read Only	
10001	Push-button SW1
10002	Push-Button SW2
10003	Push-button SW3
10004	Push-button SW4
10005	Push-button SW5
10006	Not connected – reads as 0
10007	Not connected – reads as 0
10008	Not connected – reads as 0
10009	DIP switch A2F_DIP 1
10010	DIP switch A2F_DIP 2
10011	DIP switch A2F_DIP 3
10012	DIP switch A2F_DIP 4
10013	Not connected – reads as 0
10014	Not connected – reads as 0
10015	Not connected – reads as 0
10016	Not connected – reads as 0

[1] *Note that when retargeting the reference design to the SmartFusion A2F200-EVAL-KIT board, some of the board resources mapped to Modbus registers are not available or the relevant chip-level I/O signals are connected to other resources. For example, the A2F200-EVAL-KIT board only has push-buttons SW1 and SW2 available for customer use and does not have the A2F_DIP DIP switches at all.*

Table 2-3 • **Supported Modbus Registers (continued)**

Modbus Address	Physical Resource
Input Registers – 16-Bit Read Only	
30001	RTC value in seconds (0 to 65535)
30002	ACE RV1 3.3 V pot voltage in mV
Holding Registers – 16-Bit Read/Write	
40001 ... 40038	OLED (2 line ×19 character) positions 1 .. 38 (only lower 8 bits of 16-bit value displayed but full 16 bits stored internally and read back0

Adding New Registers

New registers can be added by redefining the relevant REG_<Modbus-registerclass>_ NREGS configuration manifest constant where <Modbus-register-class> is one of DISCRETE, COILS, INPUT or HOLDING. Once defined, the slave allocates sufficient buffer memory for storing the registers.

Connecting these registers up to hardware board resources requires the modification of the relevant FreeModbus register access handler callback function eMBRegDiscreteCB(), eMBRegCoilsCB(), eMBRegInputCB(), or eMBRegHoldingCB() as well as the possible modification of the Libero SoC hardware project to support the necessary hardware resources.

Adding Support for Additional Modbus Function Codes

By default, the reference design slave provides support for the following Modbus function codes:

- Read Input Register (function code 0x04)
- Read Holding Registers (0x03)
- Write Single Register (0x06)
- Write Multiple Registers (0x10)
- Read/Write Multiple Registers (0x17)
- Read Coils (0x01)
- Write Single Coil (0x05)
- Write Multiple Coils (0x0F)
- Read Discrete Inputs (0x02)
- Report Slave ID (0x11)

See the FreeModbus API documentation [Reference 7] for information about adding support for other Modbus functions by adding callbacks to handle the relevant function codes.

3 – Libero SoC Hardware Project

A Libero SoC v10.0 SPB project using the SmartFusion MSS v2.5.106 microcontroller subsystem (MSS) is provided which implements the hardware design on which the reference design slave firmware runs.

MSS Resources Used

The reference design Libero SoC hardware project uses the following SmartFusion MSS resources by default:

1. Clock configuration: all clocks (FCLK, ACLK, PCLK0, PCLK1) are 50 MHz, derived from the 100 MHz on-chip RC oscillator.
2. Serial communications: UART_0 for RS-232 and UART_1 for RS-485 communications on the A2F500-DEV-KIT board.
3. Timers:
   - Timer 1: Used to generate a 50 µs timer interrupt required by FreeModbus for Modbus protocol timing.
   - Timer 2: Used in UART_1/RS-485 communications mode to implement an 8 ms delay for MAXIM MAX3430 RS-485 transceiver transmit/receive turnaround timing and to allow for appropriate settling time when toggling the transceiver's drive enable/receive enable – DE/REn signals. If RS-485 mode is not being used, MSS Timer 2 is freed up for other use.

In addition to the resources above required for the core Modbus functionality, the following resources are used to implement the demo program and reference design slave Modbus registers:

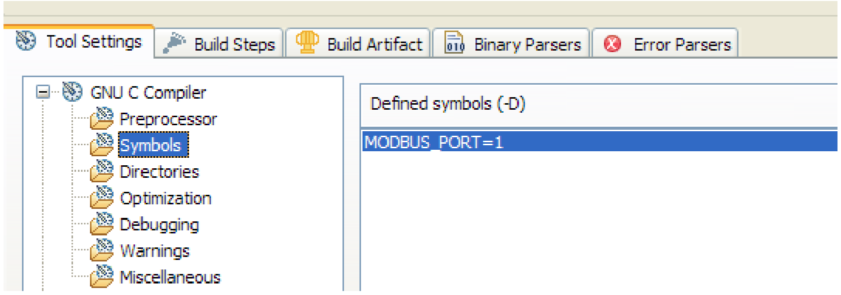
4. ARM® Cortex™-M3 SysTick: Used by the demo program to generate a 100 ms timer interrupt, from which the ISR is used to synchronize board hardware resources and reference design slave Modbus registers.
5. ACE: Used to implement the RV1 3.3 V pot Modbus input register.
6. RTC: Used to implement the RTC Modbus register.
7. I2C_0: Used to interface to the OLED display for displaying the reference design slave holding registers.
8. GPIOs: Used to interface to LEDs (x4), push-buttons (x5) and DIP switches (x4), which are used to implement the reference design slave Modbus discrete input and coil registers.

Adapting the Hardware Design

For the most part, the only changes that might need to be made to the hardware design are those required in order to map additional hardware resources to Modbus slave registers. The reference design slave comes with a set of illustrative Modbus registers mapped to specific hardware board resources.

4 – RS-485 Communications

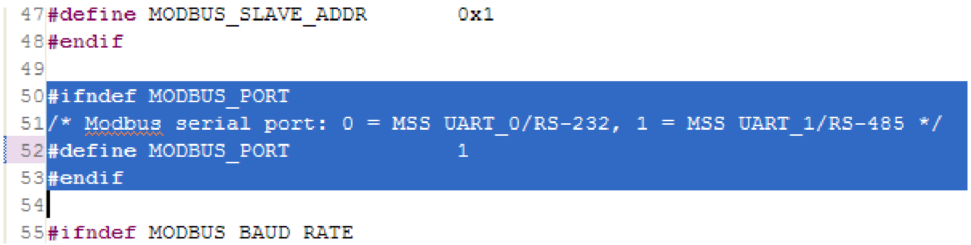
By default, the reference design firmware SoftConsole project is configured to use MSS UART_0/RS-232 communications. To use MSS UART_1/RS-485 communications on the A2F500-DEV-KIT board, the firmware can easily be reconfigured by defining the MODBUS_PORT manifest constant to be 1 instead of 0. This can be done via the project properties (**Properties** > **C/C++ Build** > **Settings** > **GNU C Compiler** > **Symbols**), as shown in Figure 4-1, or the manifest constant can be modified as shown in Figure 4-2 by editing Modbus_MSS_MSS_CM3_0_app/demo/Microsemi_SmartFusion/demo.c.



Defined symbols (-D)

MODBUS_PORT=1

Figure 4-1 **• Define MODBUS_PORT in Project Properties**



```
#define MODBUS_SLAVE_ADDR       0x1
#endif

#ifndef MODBUS_PORT
/* Modbus serial port: 0 = MSS UART_0/RS-232, 1 = MSS UART_1/RS-485 */
#define MODBUS_PORT             1
#endif

#ifndef MODBUS_BAUD_RATE
```

Figure 4-2 **• Define MODBUS_PORT in demo.c**

For RS-485 communications, an appropriate RS-485 based master and RS-485 A/B (also known as D+/D-) plus GND differential encoding twisted pair network cabling is required; optionally bus/multi-drop (multiple slaves connected to a single master) rather than point to point (master to single slave). See [Reference 4] and [Reference 10] for more details about RS-485 network cabling.

When using a PC hosted Modbus master to exercise the reference design slave, an RS-232 to RS-485 converter dongle is usually required since PCs normally do not come with an RS-485 port by default. Examples of such dongles used during the development of the reference design include these:

- DealExtreme.com: www.dealextreme.com/p/rs232-to-rs485-converter-6040
- FocalPrice.com: www.focalprice.com/CN051B/Data_Communication_Product_RS232RS485_Converter_Black.html

When using Modbus over RS-485, appropriate and timely control of the transceiver's drive/receive enable signals for half duplex transmit/receive turnaround is critical. Refer to this Netrino® article for some background on this in a general (non Modbus specific) context:

www.netrino.com/Embedded-Systems/How-To/RS-485-Transmit-Enable-Signal

In the reference design, the A2F500-DEV-KIT board's MAXIM MAX3430 RS-485 transceiver transmit/receive (DE/REn –drive/receive enable) signals are managed by the firmware using the MSS UART_1 modem control RTSn (inverted) and DTRn signals and the MSS Timer 2 is used to allow for settling time on these signals. The approach taken to transmit/receive turnaround management corresponds to option 5 in the Netrino article.

A – References

1. Microsemi SoC Products Group (formerly Actel) System Solutions home page:

   www.microsemi.com/soc/products/solutions/default.aspx
2. Microsemi SmartFusion cSoC home page:

   www.microsemi.com/soc/products/smartfusion/default.aspx
3. FAQ: www.modbus.org/faq.php

   The Modbus Organization home page: http://www.modbus.org
4. Technical resources including specifications and links to free and commercial Modbus tools and resources: www.modbus.org/tech.php
5. Wikipedia page on Modbus: en.wikipedia.org/wiki/Modbus
6. FreeModbus home page: freemodbus.berlios.de/
7. API documentation: freemodbus.berlios.de/api/index.html
8. Examples usage using Modpoll: freemodbus.berlios.de/index.php?idx=1
9. Selected suggested Modbus master tools for testing and exercising the reference design:

   proconX Pty Ltd Modpoll® a freeware (www.modbusdriver.com/info/LICENSE-FREE) PC hosted command line read-only Modbus master: www.modbusdriver.com/modpoll.html

   Automated Solutions Inc Modbus RTU/ASCII Master ActiveX Control and example programs: www.automatedsolutions.com/demos/#MBACTIVEX. A 30 day trial demo version is available for download from Automated Solutions Inc.

   www.automatedsolutions.com/products/modbusrtu.asp
10. Modbus tutorials and overviews

    Automation.com™ Introduction to Modbus: www.automation.com/resources-tools/articles-white-papers/fieldbus-serial-bus-io-networks/introduction-to-modbus

    National Instruments™ Introduction to Modbus: zone.ni.com/devzone/cda/tut/p/id/7675

    AutomatedBuildings.com Introduction to the Modbus Protocol

    Part 1:

    www.automatedbuildings.com/news/sep08/articles/cctrls/080819014909cctrls.htm

    Part 2:
    www.automatedbuildings.com/news/dec08/articles/cctrls/081124120101cctrls.htm

B – List of Changes

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
Revision 3 (January 2013)	Added "Board Settings" section (SAR 43469).	8
Revision 2 (May 2012)	Modified "Installing the Reference Design" section (SAR 38392)	7
	Replaced Figure 1-1 (SAR 38392)	7
	Replaced the command in "Read Coils Registers" (SAR 38392)	13
	Modified Table 2-1 (SAR 38392)	20
Revision 1 (January 2012)	Modified the "Installing the Reference Design" section (SAR 36029).	7
	Modified the "Read Holding Registers" section (SAR 36029).	12
	Modified the "Using Automated Solution's MiniHMI"section (SAR 36029).	13
	Modified the "Project Layout" section (SAR 36029).	19
	Modified the "Slave Firmware Configuration #defines" section (SAR 36029).	21
	Modified the "Linker Scripts" section (SAR 36029).	22
	Modified the text listed under "RS-485 Communications" section (SAR 36029).	27
	Changed all the references of Libero IDE to Libero SoC (SAR 36029).	

Note: _*The part number is located on the last page of the document. The digits following the slash indicate the month and year of publication._

C – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.