SmartFusion2 Modbus Reference Design

User Guide

Microsemi®

Table of Contents

1 – SmartFusion2 Modbus Reference Design

Introduction

Modbus is a serial communications and application level protocol primarily targeting industrial network communications. The origins of rationale for and specifications related to Modbus are summarized in the Modbus Organization FAQ [reference 3.1], the Modbus organization technical resources page [reference 3.2], and also in various Modbus tutorials and introductory guides available on the web such as those mentioned in "References" section on page 21 [reference 7].

Change Log

- V1.0 - Initial release using the Libero® System-on-Chip (SoC) v11.0 Beta SP1, SmartFusion®2 microcontroller subsystem (MSS) 0.0.720, and SoftConsole v3.4[.0.3] Beta.
- V1.1 - Updated release using the Libero SoC v11.0 Production, SmartFusion2 MSS 1.0.100, and SoftConsole v3.4[.0.5] production.

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 v1.5 [reference 5] targets the Microsemi SmartFusion2 system-on-chip (SoC) field programmable gate array (FPGA) [reference 2] SF2-DEV-KIT board (by default targeting RevB/ES/M2S050T_ES896FBGA device).
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 multi-drop master and multiple slaves—half-duplex 2/3 wire bus) physical communication mediums.
4. Includes the complete Libero SoC (v11.0/SmartFusion2 MSS v1.0.100) hardware project, which includes the SoftConsole (v3.4[.0.5]) firmware workspace and applications or library projects implementing the reference design Modbus slave which can be exercised using any third party Modbus master (including the PC hosted ones mentioned in [reference 6]) and which can be adapted and extended for customer specific requirements.
5. Supports Modbus functions. Being based on the FreeModbus communications stack the reference design supports the following Modbus functions out of the box:
   - Read Input Register (function code 0x04)
   - Read Holding Registers (function code 0x03)
   - Write Single Register (function code 0x06)
   - Write Multiple Registers (function code 0x10)
   - Read/Write Multiple Registers (function code 0x17)
   - Read Coils (function code 0x01)
   - Write Single Coil (function code 0x05)
   - Write Multiple Coils (function code 0x0F)
   - Read Discrete Inputs (function code 0x02)
   - Report Slave ID (function code 0x11)

Refer to the FreeModbus API documentation [reference 5.1] for information about extending the slave to support additional Modbus function codes.

6. Supports a variety of 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 memory (holding), light emitting diode (LED)s coils, dual in-line package (DIP) switches (discrete inputs), pushbuttons (discrete inputs), real time counter (RTC), and tick counter (inputs). The number and type of registers can be extended by the end user.

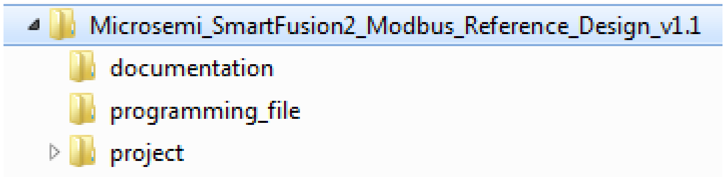
Installing and Using the Reference Design

Installing the Reference Design

Design files are available for download at:

http://soc.microsemi.com/download/rsc/?f=SF2_Modbus_Reference_DF

This reference design is delivered as a **.zip** file, whose contents should can be extracted to a suitable folder on disk. Once extracted the following folders are available as shown in Figure 1-1.



Microsemi_SmartFusion2_Modbus_Reference_Design_v1.1
documentation
programming_file
project

***Figure 1-1* • Folder Contents**

Table 1-1 briefly explains the contents of each folder.

***Table 1-1* • Folder Contents**

Folder	Description
`documentation`	Contains documentation relating to the reference design - including this user's guide.
`programming_file`	Contains programming files for the reference design with the default configuration outlined. The programming files target the SF2-DEV-KIT board. Two different programming files are provided for the two different devices that can be present on the SF2-DEV-KIT board: one for RevB/ES/M2S050T_ES 896FBGA devices (modbus_sf2_dev_kit_m2s050t_es_896fbga.stp) and one for RevC/PP/M2S050T 896FBGA devices (modbus_sf2_dev_kit_m2s050t_896fbga.stp). It is critical to use the correct STAPL file for the relevant device.
`project`	Contains the Libero SoC v11.0 project for the reference design. It also contains the SoftConsole v3.4 firmware workspace and projects in the project folder structure.

To get started with the reference design, follow these steps:

1. Run **FlashPro v11.0** standalone and program `...\Microsemi_SmartFusion2_Modbus_Reference_Design_v1.0\programming_file\modbus_top.stp` to program the SF2-DEV-KIT board.

   or

   Run **Libero SoC v11.0** and open `...\Microsemi_SmartFusion2_Modbus_Reference_Design_v1.0\project\sf2_modbus.prjx`
2. If necessary bring the design through the design flow and program the SF2-DEV-KIT board from within Libero SoC.

3. Note that while using the project, care should be taken to reconfigure the SmartFusion2 MSS ENVM data storage client named **Firmware** in order to point it at the reference design firmware Intel Hex file which is located here (if required):
   `...\Microsemi_SmartFusion2_Modbus_Reference_Design_v1.0\project\SoftConsole\modbus_MSS_CM3\modbus_MSS_CM3_app\Release\modbus_MSS_CM3_app.hex`

Power cycle the SF2-DEV-KIT board and we can see LED8 blinking every second to indicate that the Modbus reference design was programmed successfully and the firmware is running correctly. Use a suitable Modbus master utility so that the information provided in the following sections would help in interacting with the Modbus reference design.

Default Communication Settings

The default communication settings are as follows:

- Modbus serial RTU mode.
- Modbus slave address 0x01.
- SmartFusion2 MSS MMUART_1/RS-232 physical layer communications.
- 19200 baud rate.
- 8 data bits - as required by Modbus RTU mode. If the firmware is reconfigured to run in ASCII mode then the Modbus master must be configured to use 7 data bits as required by Modbus ASCII mode.
- Even parity.
- 1 stop bit.
- While using MSS MMUART_1/RS-232 the following jumpers must be installed:
  – J197:1-2 (RS232_ROUT)
  – J188:1-2 (RS232_DIN)
- While connecting a Modbus master to the reference design slave connect a serial cable from the PC to the SF2-DEV-KIT board's MSS MMUART_1 J198 DB9_RS232 connector. Note down the port number that is used on the PC configuring the master Modbus and communication settings.

If the firmware is used to reconfigure MSS MMUART_0/RS-485, then please refer to "RS-485 Communications" section on page 19.

Using Modpoll

Modpoll® is a simple command line read-only freeware Modbus master available from proconX Pty Ltd. [reference 6.1]. 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 a 16-bit representation of the firmware's 100ms systick counter, use the following command:

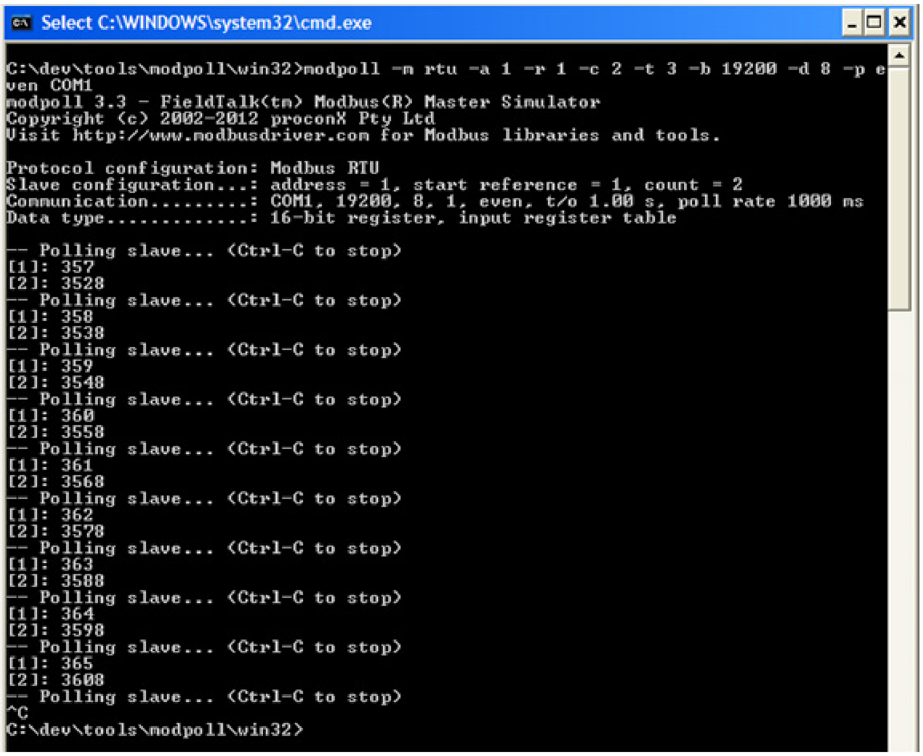
```
modpoll.exe -m rtu -a 1 -r 1 -c 2 -t 3 -b 19200 -d 8 -p even COM1
```

The command line parameters are listed in Table 1-2.

Table 1-2 **• Command Line Parameters**

Command Line Options	Description
`-m rtu`	Use RTU 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) from which it would start reading.
`-c 2`	Number of values to poll.
`-t 3`	Poll 16-bit read-only input registers - "Reference Design Slave Modbus Register Map" section on page 17 provides information about the Modbus register map supported by the reference design slave.
`-b 19200`	Baud rate.
`-d 8`	RTU mode uses 8 data bits. ASCII mode uses 7 data bits.
`-p even`	Even parity.
`COM1`	The PC COM port for connecting to the SF2-DEV-KIT board. Modify this according to your local setup.

Modpoll continuously polls the two 16-bit input registers implemented by the reference design slave: RTC 1 second counter and 100ms systick counter. You can see the first register (RTC) counting up in seconds and the second register (100ms systick counter) counting up in 1/10th of a second.



```
C:\dev\tools\modpoll\win32>modpoll -m rtu -a 1 -r 1 -c 2 -t 3 -b 19200 -d 8 -p e
ven COM1
modpoll 3.3 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: address = 1, start reference = 1, count = 2
Communication.........: COM1, 19200, 8, 1, even, t/o 1.00 s, poll rate 1000 ms
Data type.............: 16-bit register, input register table

-- Polling slave... (Ctrl-C to stop)
[1]: 357
[2]: 3528
-- Polling slave... (Ctrl-C to stop)
[1]: 358
[2]: 3538
-- Polling slave... (Ctrl-C to stop)
[1]: 359
[2]: 3548
-- Polling slave... (Ctrl-C to stop)
[1]: 360
[2]: 3558
-- Polling slave... (Ctrl-C to stop)
[1]: 361
[2]: 3568
-- Polling slave... (Ctrl-C to stop)
[1]: 362
[2]: 3578
-- Polling slave... (Ctrl-C to stop)
[1]: 363
[2]: 3588
-- Polling slave... (Ctrl-C to stop)
[1]: 364
[2]: 3598
-- Polling slave... (Ctrl-C to stop)
[1]: 365
[2]: 3608
-- Polling slave... (Ctrl-C to stop)
^C
C:\dev\tools\modpoll\win32>
```

Figure 1-2 **• Read Input Registers**

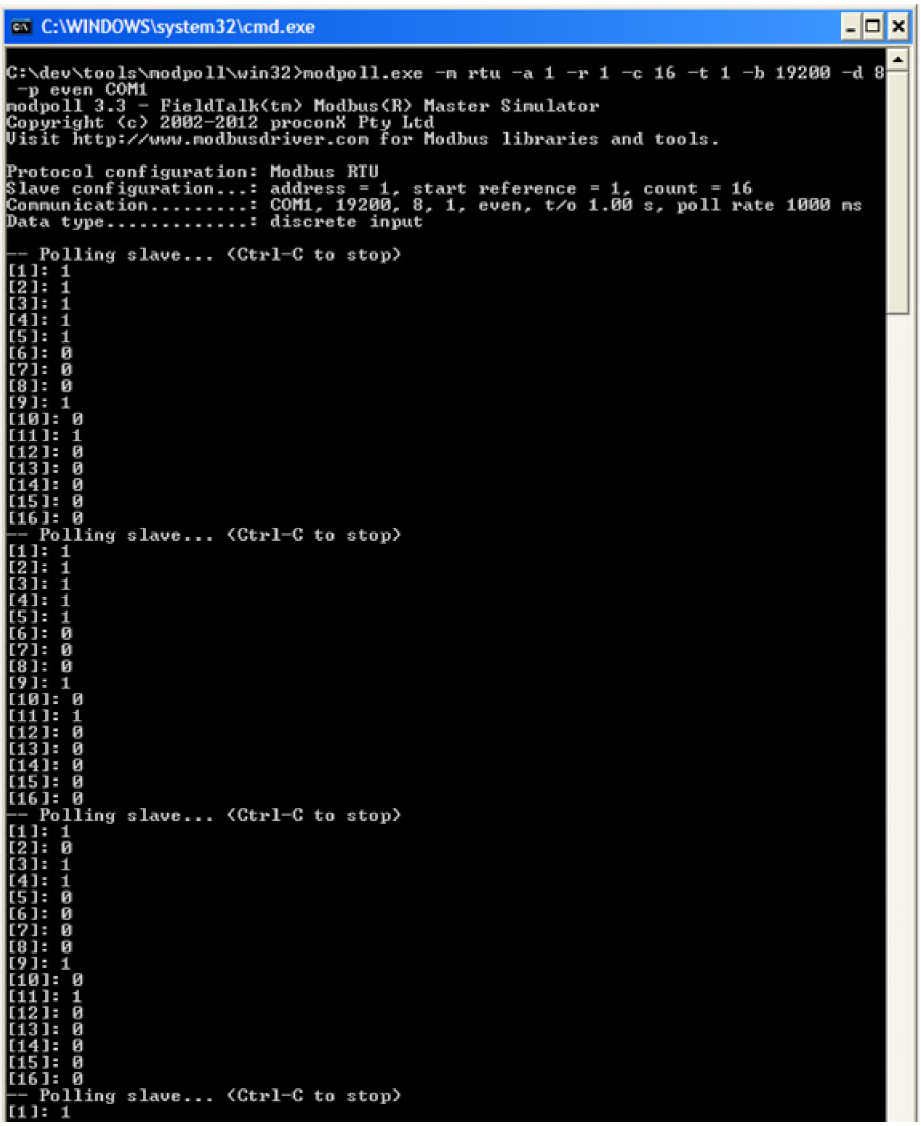
Read Discrete Input Registers

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 COM1
```

This time use `-t 1 -c 16` to read the 16 single bit read-only discrete inputs supported by the reference design slave. "Reference Design Slave Modbus Register Map" section on page 17 provides information about the Modbus register map supported by reference design slave.

As Modpoll polls the slave, toggle the SF2-DEV-KIT board's SW1-SW5 pushbuttons and SW10 DIP switches to see the effect that it has on the results reported by Modpoll.



```
C:\dev\tools\modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 16 -t 1 -b 19200 -d 8
 -p even COM1
modpoll 3.3 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: address = 1, start reference = 1, count = 16
Communication.........: COM1, 19200, 8, 1, even, t/o 1.00 s, poll rate 1000 ms
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]: 1
[10]: 0
[11]: 1
[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]: 1
[10]: 0
[11]: 1
[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]: 0
[6]: 0
[7]: 0
[8]: 0
[9]: 1
[10]: 0
[11]: 1
[12]: 0
[13]: 0
[14]: 0
[15]: 0
[16]: 0
-- Polling slave... (Ctrl-C to stop)
[1]: 1
```

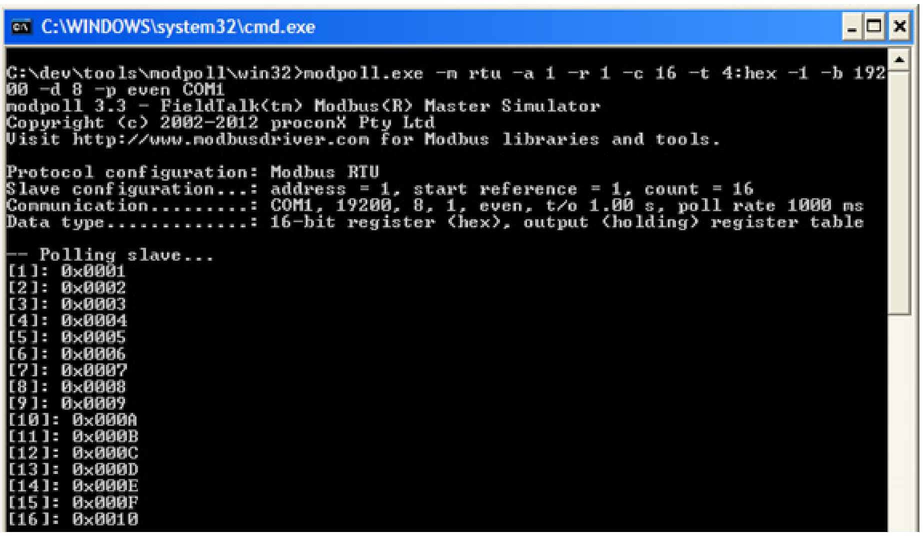
Figure 1-3 • **Read Discrete Input Registers**

Read Holding Registers

The reference design implements 16x16 bit holding registers stored in RAM (and hence volatile). The firmware initializes them to the values 0x01 to 0x10 at start up and thereafter they can be read or written. Modpoll is a read-only Modbus master. To read these registers, run Modpoll using the following command:

```
modpoll.exe -m rtu -a 1 -r 1 -c 16 -t 4:hex -1 -b 19200 -d 8 -p even COM1
```

In this case `-c 16` tells Modpoll to read 16 registers, `-t 4:hex` tells it to read the 16-bit holding registers and display them in hex and `-1` tells it to poll once rather than continuously.



```
C:\dev\tools\modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 16 -t 4:hex -1 -b 192
00 -d 8 -p even COM1
modpoll 3.3 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: address = 1, start reference = 1, count = 16
Communication.........: COM1, 19200, 8, 1, even, t/o 1.00 s, poll rate 1000 ms
Data type.............: 16-bit register (hex), output (holding) register table

-- Polling slave...
[1]: 0x0001
[2]: 0x0002
[3]: 0x0003
[4]: 0x0004
[5]: 0x0005
[6]: 0x0006
[7]: 0x0007
[8]: 0x0008
[9]: 0x0009
[10]: 0x000A
[11]: 0x000B
[12]: 0x000C
[13]: 0x000D
[14]: 0x000E
[15]: 0x000F
[16]: 0x0010
```

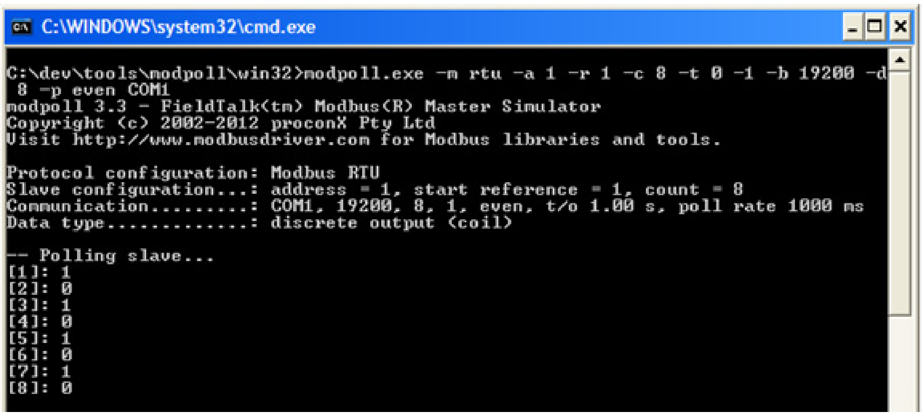
Figure 1-4 • **Read Holding Registers**

Read Coils Registers

To read the 8 1-bit read-write coils registers (7 of which are connected to the SF2-DEV-KIT board's LEDs) run Modpoll using the following command:

```
modpoll.exe -m rtu -a 1 -r 1 -c 8 -t 0 -1 -b 19200 -d 8 -p even COM1
```

Note that the LEDs are active Low, so a ’0’ means that the LED is ON, while ’1’ means that it is OFF.



```
C:\dev\tools\modpoll\win32>modpoll.exe -m rtu -a 1 -r 1 -c 8 -t 0 -1 -b 19200 -d
 8 -p even COM1
modpoll 3.3 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: address = 1, start reference = 1, count = 8
Communication.........: COM1, 19200, 8, 1, even, t/o 1.00 s, poll rate 1000 ms
Data type.............: discrete output (coil)

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

Figure 1-5 • **Read Coils Registers**

Using Automated Solutions Inc's MiniHMI

Automated Solutions Inc [reference 6.2] 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. Please contact Automated Solutions Inc or refer to their website for details of their commercial tool offerings and prices.

Automated Solutions Inc. also provide a free fully featured 30 day trial version of the Modbus RTU/ASCII Master ActiveX Control and example applications which can be used to exercise and demonstrate the features of the Microsemi SmartFusion2 Modbus reference design slave.

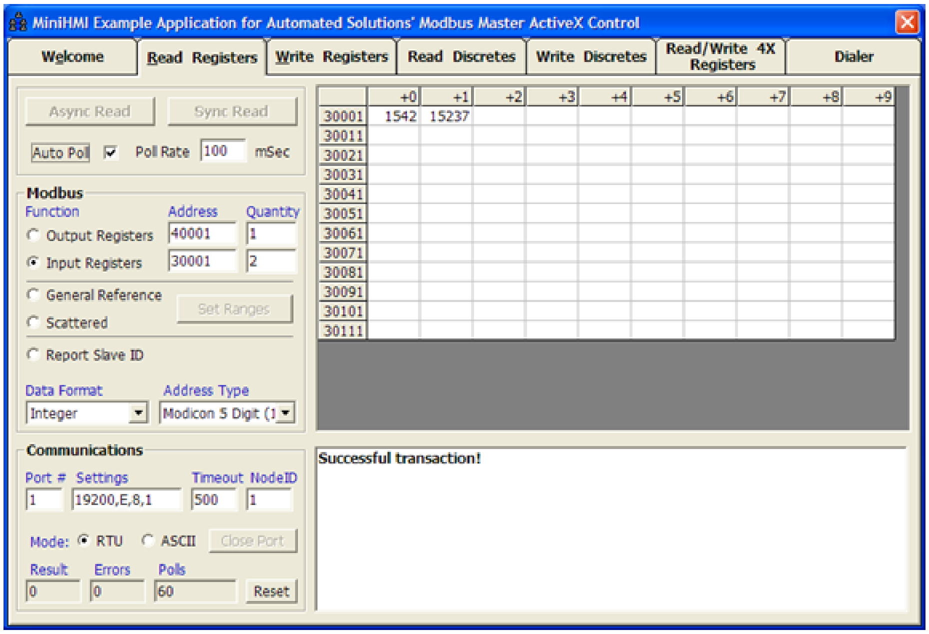
Obtain and install the demo or full version of the Automated Solutions Inc. Modbus RTU/ASCII Master ActiveX Control package.

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 on the **Read Registers** tab. Ensure that the **Communications** settings are configured appropriately to match your the slave setup. For default settings refer to "Default Communication Settings" section on page 7. The MiniHMI settings with the possible exception of Communications Port match the reference design slave default settings. Select the **Modbus** > **Function** > **Input Registers** radio button and in the Quantity field enter 2. Check the **Auto Poll** check box and MiniHMI should start continuously polling the one second RTC and 100ms systick counter input registers whose values are updated as time passes. Uncheck the **Auto Poll** check box to stop continuous polling.



MiniHMI Example Application for Automated Solutions' Modbus Master ActiveX Control

Auto Poll ☑ Poll Rate 100 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
1	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII

Result	Errors	Polls
0	0	60

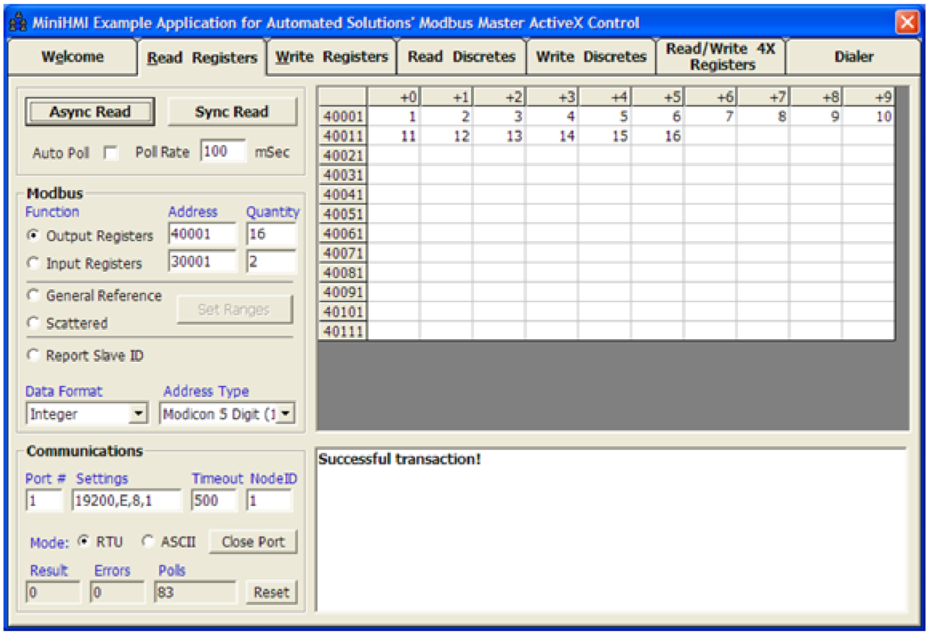
	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
30001	1542	15237								
30011										
30021										
30031										
30041										
30051										
30061										
30071										
30081										
30091										
30101										
30111										

Successful transaction!

Figure 1-6 • **MiniHMI Example Application**

Read/Write Holding Registers

On the Read Registers tab select the **Modbus** > **Function** > **Output Registers** radio button and in the **Quantity** field enter 16. Click the **Async Read** or **Sync Read** button and MiniHMI should read back the 16 RAM based holding registers.



Auto Poll □ Poll Rate 100 mSec

Modbus

Function	Address	Quantity
● Output Registers	40001	16
○ Input Registers	30001	2

○ General Reference
○ Scattered
○ Report Slave ID

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

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
40001	1	2	3	4	5	6	7	8	9	10
40011	11	12	13	14	15	16				
40021										
40031										
40041										
40051										
40061										
40071										
40081										
40091										
40101										
40111										

Communications

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

Mode: ● RTU ○ ASCII

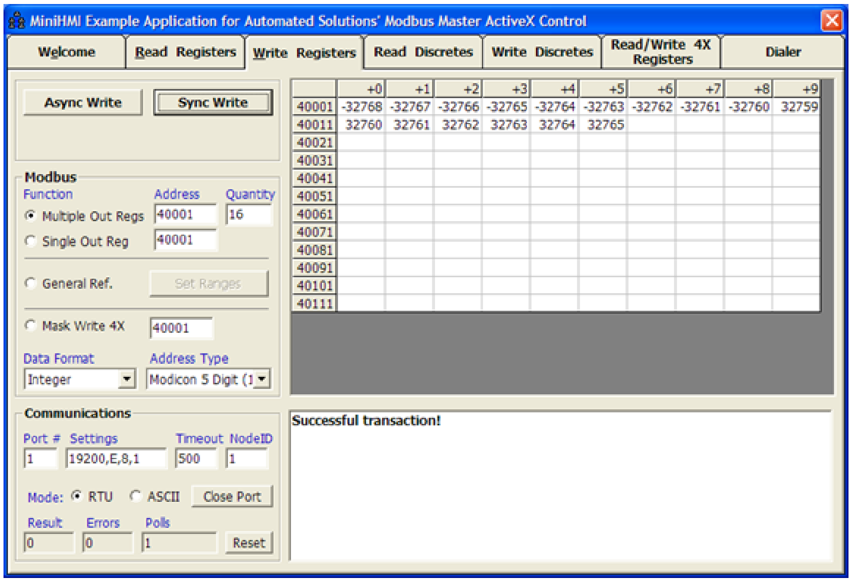
Result	Errors	Polls
0	0	83

Successful transaction!

Figure 1-7 **• MiniHMI Example Application - Async Read**

To write to the holding registers select the **Write Registers** tab. As above shown in Figure 1-7 ensure that the **Communications** settings are configured correctly. Note that the **Communications** settings are 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 16 in the **Quantity** field according to the number of many registers to be written. In the register grid/spreadsheet view enter the number of values to be written to the holding registers. Click the **Async Write** or **Sync Write** to flush the new values to the reference design Modbus slave. Note that the 16-bit values are treated as signed so the values must be between xxx and -32768 and +32767.



Modbus

Function	Address	Quantity
● Multiple Out Regs	40001	16
○ Single Out Reg	40001	

○ General Ref.
○ Mask Write 4X 40001

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

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
40001	-32768	-32767	-32766	-32765	-32764	-32763	-32762	-32761	-32760	32759
40011	32760	32761	32762	32763	32764	32765				
40021										
40031										
40041										
40051										
40061										
40071										
40081										
40091										
40101										
40111										

Communications

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

Mode: ● RTU ○ ASCII

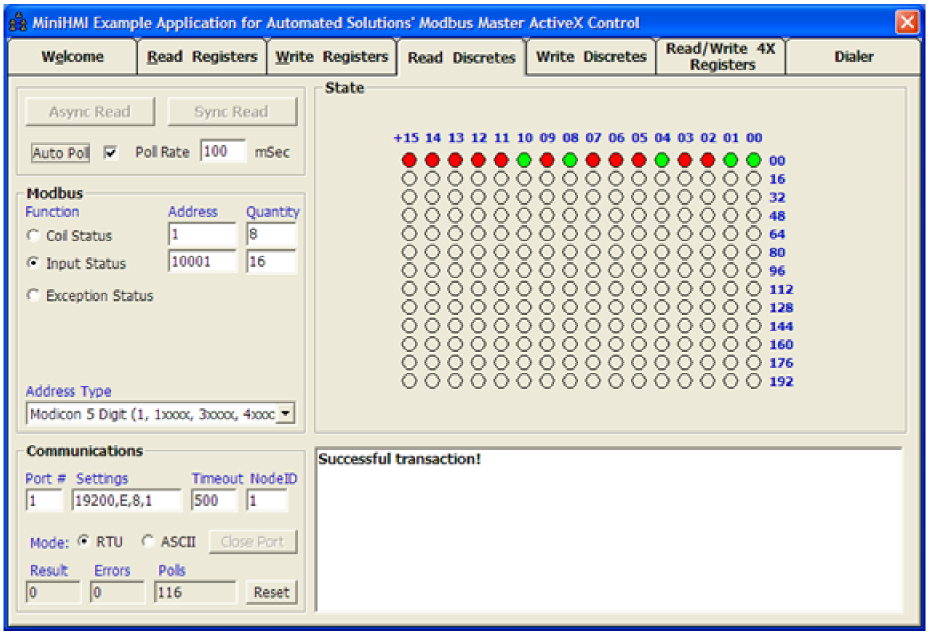
Result	Errors	Polls
0	0	1

Successful transaction!

Figure 1-8 **• MiniHMI Example Application - Sync Write**

Read Discrete Inputs/Coils

To read the discrete inputs and coils go to the **Read Discretes** tab, ensure that the **Communications** settings are set correctly, choose to read 8 or fewer coils or 16 or fewer discrete inputs and then click **Async Read** or **Sync Read** to read once or Auto Poll to poll continuously. While reading the 16 discrete inputs the **MiniHMI GUI's State view** reflects the changes due to manual toggling of the SF2-DEV-KIT board's SW1-5 pushbuttons or SW10 DIP switches.



☑ Auto Poll   Poll Rate 100 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
1	19200,E,8,1	500	1

Mode: ● RTU ○ ASCII

Result	Errors	Polls
0	0	116

State

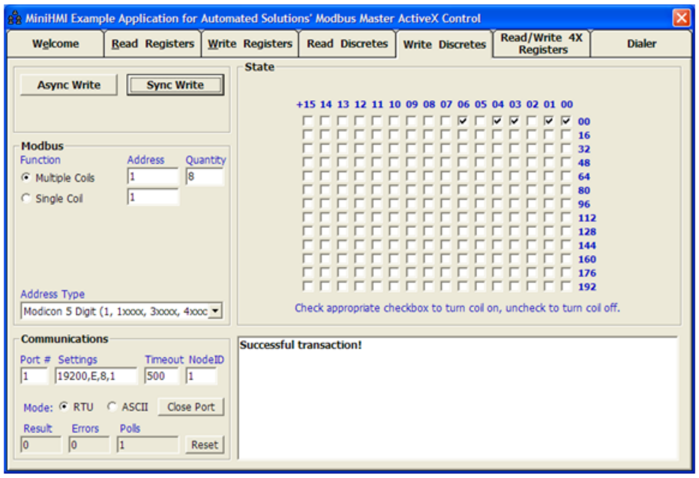
+15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
🔴	🔴	🔴	🔴	🔴	🟢	🔴	🟢	🔴	🔴	🔴	🟢	🔴	🔴	🟢	🟢	00
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	16
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	32
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	48
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	64
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	80
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	96
○	○	○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

Successful transaction!

Figure 1-9 • **MiniHMI Example Application - Read Discretes**

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 8 (only coils 1-7 are actually connected to board resources/LEDs by default), toggle any of the first seven coils (00-06) in the State view and then clock the **Async Write** or **Sync Write** to change the status of the LEDs. You should see the changes reflected on the SF2-DEV-KIT board.



Modbus

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

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

Communications

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

Mode: ● RTU ○ ASCII

Result	Errors	Polls
0	0	1

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.

Successful transaction!

Figure 1-10 • **MiniHMI Example Application - Write Discretes**

Using Other Modbus Masters

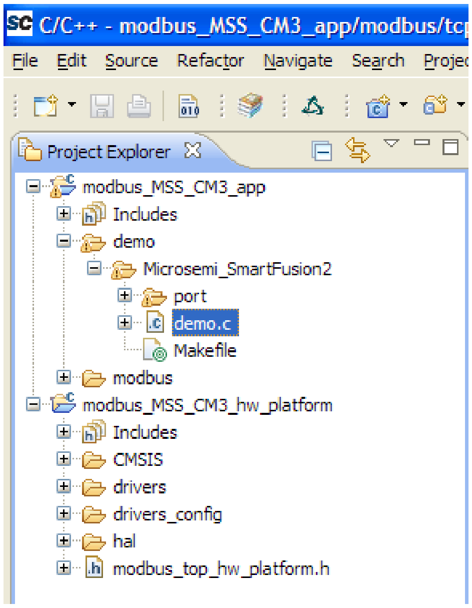
Similar to the proconX Pty Ltd Modpoll and Automated Solutions Inc Modbus RTU/ASCII ActiveX Component MiniHMI example applications can be used to interact with the Microsemi SmartFusion2 reference design sample Modbus slave, any other Modbus compatible PC hosted or other master can also be used. Ensure that the master Modus and serial communications settings match the slave target settings.

SoftConsole Firmware Project

The SoftConsole firmware workspace and projects are bundled with the Libero SoC v11.0 project because they are created using the Libero embedded firmware flow. Open the reference design project in Libero SoC v11.0, go to **Project** > **Tool Profiles...** and ensure that the **Tools** > **Software IDE** tool profile is configured correctly to run SoftConsole v3.4. Then, right-click on **Design Flow** > **Develop Firmware** > **Write Application Code** and choose **Open Interactively**. SoftConsole should launch and open the reference design SoftConsole workspace.

Project Layout

Figure 1-11 outlines the structure of the firmware project.


SC C/C++ - modbus_MSS_CM3_app/modbus/tc
File Edit Source Refactor Navigate Search Projec
Project Explorer

```
modbus_MSS_CM3_app
    Includes
    demo
        Microsemi_SmartFusion2
            port
            demo.c
            Makefile
    modbus
modbus_MSS_CM3_hw_platform
    Includes
    CMSIS
    drivers
    drivers_config
    hal
    modbus_top_hw_platform.h
```

Figure 1-11 **• Directory Structure of the Firmware Project**

When Libero generates the firmware for a project the workspace contains two projects - an "app" project (`modbus_MSS_CM3_app`) for the main application and a library project (`modbus_MSS_CM3_hw_platform`) for the common firmware drivers relevant to the target hardware configuration. Whenever the hardware design is modified and regenerated Libero updates the latter project to keep it in sync with the target hardware.

Table 1-3 summaries the contents and purpose of each folder.

***Table 1-3* • Folder Contents**

Folder/File	Description
modbus_MSS_CM3_app	The Modbus reference design application project.
modbus_MSS_CM3_app/Includes	Project level include files/folders.
modbus_MSS_CM3_app/demo/Microsemi_SmartFusion2/port	FreeModbus porting layer files targeting Microsemi SmartFusion2 device.
modbus_MSS_CM3_app/demo/Microsemi_SmartFusion2/demo.c	Reference design slave implementation program file containing main(), default slave firmware configuration #defines, Modbus register and callback implementations etc.
modbus_MSS_CM3_hw_platform	Libero generated library project containing all common firmware drivers for the target hardware platform.
modbus_MSS_CM3_hw_platform/Includes	Project level include files/folders.
modbus_MSS_CM3_hw_platform/CMSIS	The CMSIS part of the SmartFusion2 CMSIS hardware abstraction layer.
modbus_MSS_CM3_hw_platform/drivers	Various SmartFusion2 firmware drivers.
modbus_MSS_CM3_hw_platform/drivers/mss_gpio	SmartFusion2 MSS GPIO driver.
modbus_MSS_CM3_hw_platform/drivers/mss_hpdma	SmartFusion2 MSS HPDMA driver - not used here.
modbus_MSS_CM3_hw_platform/drivers/mss_nvm	SmartFusion2 MSS eNVM driver - not used here.
modbus_MSS_CM3_hw_platform/drivers/mss_rtc	SmartFusion2 MSS RTC driver.
modbus_MSS_CM3_hw_platform/drivers/mss_sys_services	SmartFusion2 MSS System Services driver - not used here.
modbus_MSS_CM3_hw_platform/drivers/mss_timer	SmartFusion2 MSS Timer driver.
modbus_MSS_CM3_hw_platform/drivers/mss_uart	SmartFusion2 MSS MMUART driver.
modbus_MSS_CM3_hw_platform/drivers_config	Hardware target configuration information required for building firmware (For example clock speeds etc.)
modbus_MSS_CM3_hw_platform/hal	The HAL part of the SmartFusion2 CMSIS hardware abstraction layer.

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++ Buil**d > **Settings** > **GNU C Compiler** > **Symbols** or by editing the `#defines` in `FreeModbus_demo/demo/Microsemi_SmartFusion/demo.c` directly before recompiling the firmware.

Table 1-4 • **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 MMUART_1/RS-485 • 1 = MSS MMUART_0/RS-232 • Default: 1
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
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: 16

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 MMUART_1/RS-232 physical layer communications
- 19200 baud rate
- 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 it set up to use the following SmartFusion2 CMSIS-HAL sample linker scripts:

- `Debug target:CMSIS/startup_gcc/debug-in-microsemi-smartfusion2-esram.ld`
- `Release target:CMSIS/startup_gcc/production-execute-in-place.ld`

The debug target supports downloading to and debugging from SmartFusion2 MSS Embedded SRAM (eSRAM). The release target creates an Intel HEX file suitable for loading into the “Firmware” MSS eNVM data storage client when while programming the board using Libero/FlashPro. Once this is done the release firmware runs from reset.

As with any SmartFusion2 firmware project other build/link and memory configurations are possible (for example, 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 etc.) but are beyond the scope of this document.

FreeModbus Configuration Options

See the FreeModbus API documentation [reference 5.1] 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 listed in Table 1-5.

***Table 1-5* • Modbus Registers**

Coils Registers – Single Bit Read-Write	
Modbus Address	Physical Resource
1	SF2-DEV-KIT LED D1
2	SF2-DEV-KIT LED D2
3	SF2-DEV-KIT LED D3
4	SF2-DEV-KIT LED D4
5	SF2-DEV-KIT LED D5
6	SF2-DEV-KIT LED D6
7	SF2-DEV-KIT LED D7
8	Not connected - reads as 0, writes are ignored. Note that SF2-DEV-KIT LED D8 is used as a “heartbeat” to indicate that the firmware is executing normally when the LED blinks on/off every 1 second.

Table 1-5 • **Modbus Registers (continued)**

Discrete Input Registers – Single Bit Read-Only	
Modbus Address	Physical Resource
10001	SF2-DEV-KIT pushbutton SW1
10002	SF2-DEV-KIT pushbutton SW2
10003	SF2-DEV-KIT pushbutton SW3
10004	SF2-DEV-KIT pushbutton SW4
10005	SF2-DEV-KIT pushbutton SW5
10006	Not connected - reads as 0
10007	Not connected - reads as 0
10008	Not connected - reads as 0
10009	SF2-DEV-KIT DIP switch SW10:1
10010	SF2-DEV-KIT DIP switch SW10:2
10011	SF2-DEV-KIT DIP switch SW10:3
10012	SF2-DEV-KIT DIP switch SW10:4
10013	Not connected - reads as 0
10014	Not connected - reads as 0
10015	Not connected - reads as 0
10016	Not connected - reads as 0
Input Registers – 16 bit Read-Only	
Modbus Address	Physical Resource
30001	RTC counter value in seconds (0 to 65535)
30002	100ms systick counter (0 to 65535)
Holding Registers – 16 bit write	
Modbus Address	Physical Resource
40001 .. 40017	16 x 16 bit read-write holding registers implemented in RAM (and hence volatile across reboots).

Adding New Registers

New registers can be added by redefining the relevant `REG_<Modbus-register-class>_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 IDE 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 (function code 0x03)
- Write Single Register (function code 0x06)
- Write Multiple Registers (function code 0x10)
- Read/Write Multiple Registers (function code 0x17)
- Read Coils (function code 0x01)
- Write Single Coil (function code 0x05)
- Write Multiple Coils (function code 0x0F)

- Read Discrete Inputs (function code 0x02)
- Report Slave ID (function code 0x11)

See the FreeModbus API documentation [reference 5.1] for information about adding support for other Modbus functions by adding callback to handle the relevant function codes. Libero IDE hardware project.

Libero SoC Hardware Project

The Libero SoC v11.0 Beta SP1 project using the SmartFusion2 MSS v0.0.720 MSS is provided. It implements the hardware design on which the reference design slave firmware runs.

MSS Resources

The reference design Libero hardware project uses the following SmartFusion2 MSS resources by default:

1. **Clock configuration**: An external 50 MHz reference clock source is used and a 100 MHz clock is derived from this to drive the Cortex-M3 processor and MSS APB buses.

   Serial communications: UART_0 for RS-232 and UART_1 for RS-485 communications on the A2F500-DEV-KIT board.
2. **Timers**
   – **Timer 1**: Used to generate a 50 us timer interrupt required by FreeModbus for Modbus protocol timing.
   – **Timer 2**: Used in MSS MMUART_0/RS-485 communications mode to implement an 20 ms delay for Texas Instruments SN65HVD12 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 then 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:

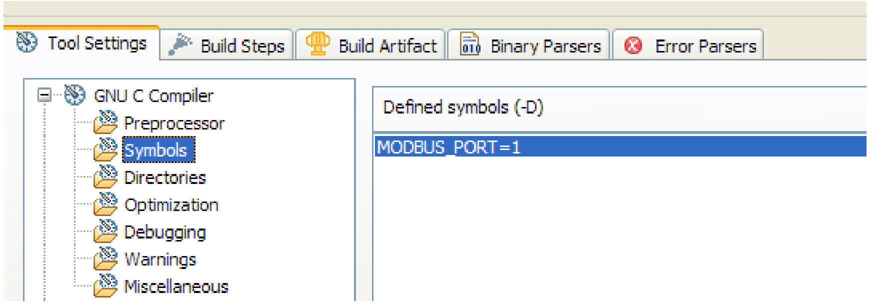
3. **Cortex-M3 SysTick**: Used by the demo program to generate a 100ms timer interrupt whose ISR is used to synchronize board hardware resources and reference design slave Modbus registers.
4. **RTC**: Used to implement the RTC Modbus register.
5. **GPIOs**: Used to interface LEDs (x8), pushbuttons (x5), 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. As explained in the previous section, the reference design slave comes with a set of illustrative Modus registers mapped to specific hardware board resources.

RS-485 Communications

By the default the reference design firmware SoftConsole project is configured to use MSS MMUART_1/RS-232 communications. To use MSS MMUART_0/RS-485 communications on the SF2-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**).

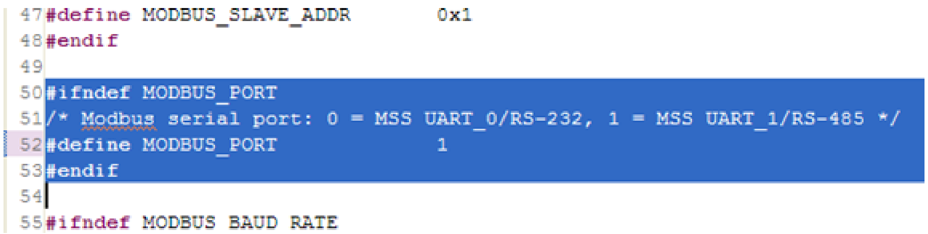


Defined symbols (-D)

MODBUS_PORT=1

***Figure 1-12* • Tool Settings - Symbols**

or by editing `.../modbus_MSS_CM3_app/demo/Microsemi_SmartFusion/demo.c` file.



```
#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 1-13* • Demo File**

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 (2/3 wire half duplex) is required - optionally bus/multi-drop (multiple slaves connected to a single master) rather than point to point (master to single slave). See [reference 3.2] and [reference 7] for more details about RS-485 network cabling.

In order to use RS-485 communications with the reference design the following jumpers must be installed on the SF2-DEV-KIT.

- J209:1-2 RS485_TE
- J178:1-2 RS485_RE
- J199:1-2 RS485_TX
- J210:1-2 RS485_RX

and the RS-485 3 wire half-duplex cable must be connected to the SF2-DEV-KIT as follows:

- RS485 GND J221:1 and/or J221:2 and/or J221:7 and/or J221:8
- RS485 D-/B J221:3 and/or J221:5
- RS485 D+/A J221:4 and/or J221:6

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: http://www.dealextreme.com/p/rs232-to-rs485-converter-6040
- FocalPrice.com: http://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 the following Netrino® article for some background on this in a general (non-Modbus specific) context:

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

In the reference design the SF2-DEV-KIT board's Texas Instruments SN65HVD12 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.

References

1. Microsemi SoC Products Group System Solutions home page: www.microsemi.com/soc/products/solutions/default.aspx
2. Microsemi SmartFusion2 SoC FPGA page: www.microsemi.com/soc/products/smartfusion2/default.aspx
3. The Modbus organization page: http://www.modbus.org
   - 3.1. FAQ: http://www.modbus.org/faq.php
   - 3.2. Technical resources including specifications and links to free and commercial Modbus tools and resources: http://www.modbus.org/tech.php
4. Wikipedia page on Modbus: http://en.wikipedia.org/wiki/Modbus
5. FreeModbus page: http://freemodbus.berlios.de/
   - 5.1. API documentation: http://freemodbus.berlios.de/api/index.html
   - 5.2. Examples usage using Modpoll: http://freemodbus.berlios.de/index.php?idx=1
6. Selected suggested Modbus master tools for testing and exercising the reference design:
   - 6.1. proconX Pty Ltd Modpoll® - a freeware (http://www.modbusdriver.com/info/LICENSE-FREE) PC hosted command line read-only Modbus master: http://www.modbusdriver.com/modpoll.html
   - 6.2. Automated Solutions Inc Modbus RTU/ASCII Master ActiveX Control and example programs: http://www.automatedsolutions.com/demos/#MBACTIVEX. A 30 day trial demo version is available for download from Automated Solutions Inc. http://www.automatedsolutions.com/products/modbusrtu.asp
7. Modbus tutorials and overviews
   - 7.1. Automation.com™ Introduction to Modbus: http://www.automation.com/resources-tools/articles-white-papers/fieldbus-serial-bus-io-networks/introduction-to-modbus
   - 7.2. National Instruments™ Introduction to Modbus: http://zone.ni.com/devzone/cda/tut/p/id/7675
   - 7.3. AutomatedBuildings.com - Introduction to the Modbus Protocol
   - 7.3.1.Part 1: http://www.automatedbuildings.com/news/sep08/articles/cctrls/080819014909cctrls.htm
   - 7.3.2.Part 2: http://www.automatedbuildings.com/news/dec08/articles/cctrls/081124120101cctrls.htm

A – 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, 408.643.6913

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.

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