



EMBEDDED MODEM DEVELOPMENT KIT USER'S GUIDE

1. Kit Contents

The Embedded Modem Development Kit contains the following items:

- C8051F120 Target Board
- AB3 Modem Adapter Board
- Si2457FT18-EVB Modem Board
- Silicon Laboratories Evaluation Kit IDE and Product Information CD-ROM. CD content includes the following:
 - Silicon Laboratories Integrated Development Environment (IDE)
 - Keil Software 8051 Development Tools (evaluation assembler, linker and 'C' compiler)
 - Source code examples and register definition files
 - Documentation
 - TCP/IP Configuration Wizard
 - Embedded Modem Development Kit User's Guide (this document)
- AC to DC Power Adapter
- USB Debug Adapter (USB to Debug Interface)
- USB Cable

2. Kit Overview

This Embedded Modem Development kit provides a means to develop real-world embedded modem applications based on the CMX Micronet™ protocol stack running on Silicon Laboratories MCU devices. See application note "AN237: TCP/IP Library Programmer's Guide" for detailed information about the TCP/IP stack. The kit includes three development boards along with the Silicon Laboratories TCP/IP Configuration Wizard and Integrated Development Environment (IDE) to expedite embedded modem designs. Section 3 takes the designer through an example illustrating each of the steps to create an application using the included software.

A C8051F120 Target Board, an AB3 Modem Adapter board, and a Si2457FT18-EVB Modem Board are included in the Embedded Modem Development kit. These boards are shown in Figure 1. The C8051F120 Target Board is a Silicon Laboratories standard development board featuring the 100 MIPS, 8051-based C8051F120 MCU. The AB3 Modem Adaptor Board connects the C8051F120 board to the Si2457 modem chipset board, the Si2457FT18-EVB.

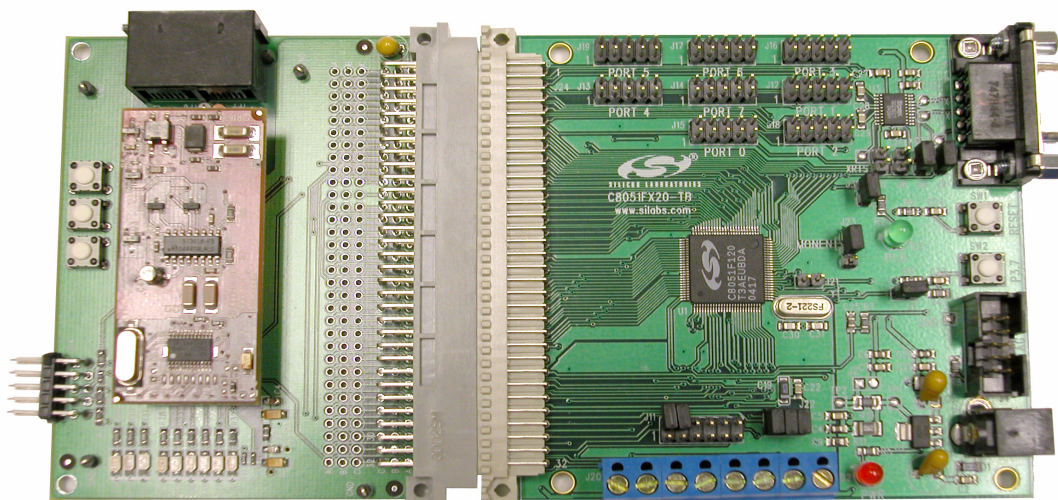


Figure 1. Embedded Modem Development Kit Hardware Overview

3. Development Kit Embedded Modem Demonstration

The example detailed below describes the steps necessary to run a simple demonstration. This example steps through hardware setup and code configuration using the TCP/IP Configuration Wizard and the Silicon Laboratories IDE to create and debug an example ping application.

3.1. Hardware Setup

Connect the hardware included with the kit according to the instructions listed below. Diagrams of the hardware setup are shown in Figure 2 and Figure 3.

1. Connect the **Si2457FT18-EVB Modem Board** to the **AB3 Modem Adapter Board** at JP1 and JP2.
2. Connect the **AB3 Modem Adapter Board** to the **C8051F120 Target Board** at J24.
3. Using the ribbon cable, connect the USB Debug Adapter to the **C8051F120 Target Board** at the JTAG header.
4. Connect one end of the provided USB cable to any available USB port on the PC.
5. Connect the other end of the USB cable into the USB connector on the USB Debug Adapter.
6. Connect the ac/dc Power Adapter to the **C8051F120 Target Board** at P1. This connection should power the boards. Power is indicated by the "PWR" LED.

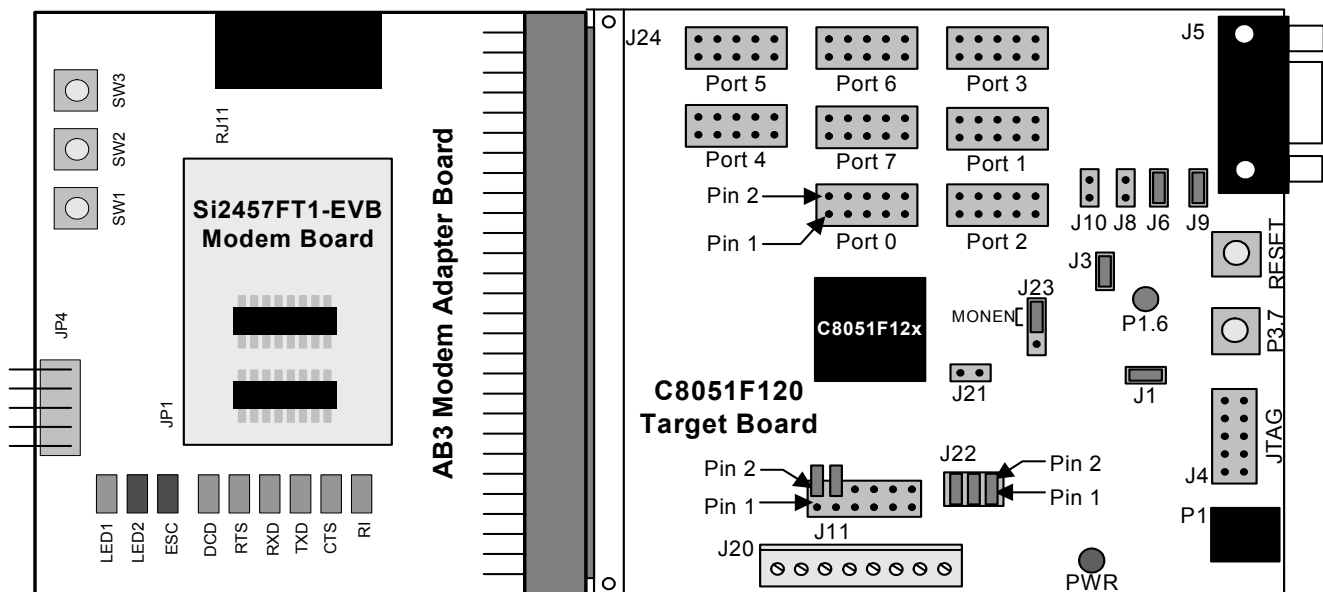


Figure 2. Embedded Modem Demo Board Connections

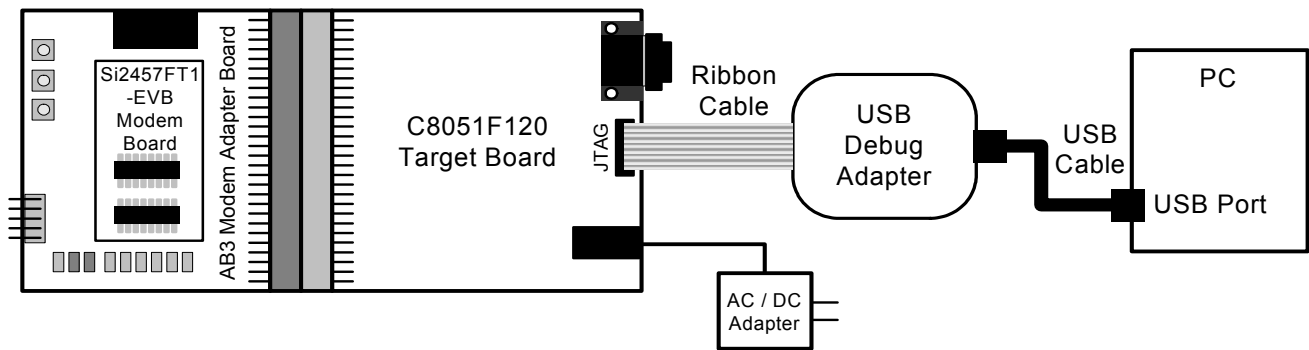


Figure 3. Embedded Modem Demo Debug Connections

3.2. Phone Line Hardware Setup

Connect the phone lines and modems according to the instructions listed below. A diagram of the phone line setup is shown in Figure 4.

1. Connect a phone cable from the target PC and modem to a phone line simulator or a phone line outlet.
2. Connect a phone cable from the **AB3 Modem Adapter Board** at the RJ11 connector to the phone line simulator or a phone line outlet.

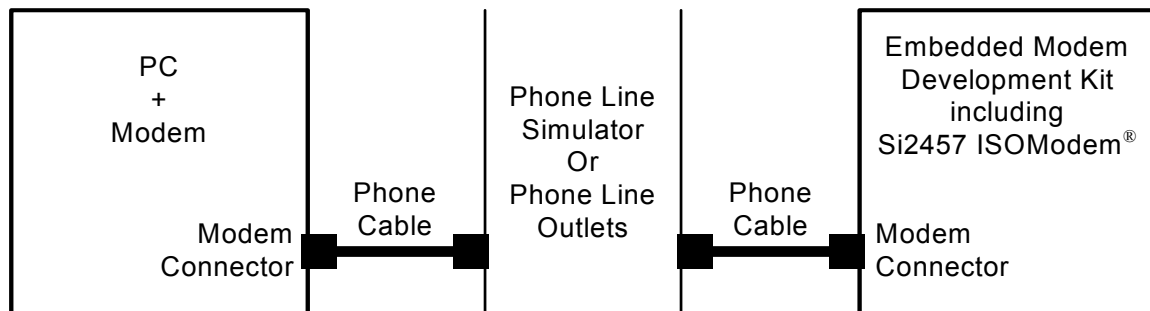


Figure 4. Embedded Modem Demo Phone Line Connections

3.3. Software Installation

Install the software included with the Embedded Modem Development kit according to the instructions below.

1. Place the Embedded Modem Development Kit CD-ROM into the PC for application install.
2. An installation dialogue box will appear. Click the **Install Development Tools** button.
3. Follow the installation prompts to install the development tools. The Silicon Laboratories IDE will be installed by default in the "C:\SiLabs\MCU" directory, the Embedded Modem examples will be installed by default in the "C:\SiLabs\MCU\Examples\Modem" directory, and the TCP/IP Configuration Wizard will be installed by default in the "C:\SiLabs\MCU\TCP-IP Config" directory. In addition, shortcuts to the applications will be placed in the **Start→Programs** menu.

3.4. Configuring PC Modem Settings

The target PC and modem must be configured to receive communications from the Si2457 in the Embedded Modem Development Kit. Follow the steps below to configure the PC.

1. Open the **Network Connections** window. Links can be found in the **Start** menu or the **Control Panel**.
2. Select **Create a New Connection** from the File menu in the **Network Connections** window.
3. The **New Connection Wizard** will open. Click the **Next** button.
4. Select the **Set up an advanced connection** option. Click the **Next** button. (This window might appear with all operating systems.)
5. Select the **Accept incoming connections** option. Click the **Next** button.
6. The next window should show all of the available devices on the PC. Check the modem that will be used for the demo.
7. The **Incoming Virtual Private Network (VPN) Connection** window does not require any selections. Click the **Next** button.
8. Check the **User** that will be allowed to connect. Take note of the **User name** and **Password** that are required for this connection. These will be needed in Section 3.6 (Step 6). Click the **Next** button.
9. Check the **Internet Protocol (TCP/IP)** option. Click the **Next** button.
10. Click the **Finish** button.

3.5. Creating Initialization Firmware with the TCP/IP Configuration Wizard

The next step is to create TCP/IP initialization firmware for the C8051F120 device. This is done using the TCP/IP Configuration Wizard included with the Embedded Modem Development Kit.

1. To run the program, select **TCP/IP Configuration Wizard** from the **Start→Silicon Laboratories** menu or run *TCPIP_Config.exe* located by default in the “C:\SiLabs\MCU\TCP-IP Config” directory.
2. In the **Hardware Settings** section in the left window, check **Si245x**.
3. In the **Settings** section in the right window, select your country from the **Country Codes** list.
4. In the **Settings** section in the right window, enter the phone number of the target modem in the **Outgoing phone number** text box.
5. In the **Protocol Settings→Link/Physical Layer** section in the left window, check **PPP** and **PAP**. No PPP or PAP setting changes are required for this demonstration.
6. In the **Protocol Settings→Transport Layer** section in the left window, check **UDP**.
7. Next, to create the firmware and project select **Generate Project** from the **File** menu.
8. When prompted for a folder to save the project, browse to the “C:\SiLabs\MCU\Examples\Modem\Demo” directory. Click the **OK** button.
9. Click the **OK** button when the the “Project generated successfully” message appears.
10. Close the TCP/IP Configuration Wizard.

3.6. Configuring the Firmware with the IDE

The next step is to customize the firmware created by the TCP/IP Configuration Wizard. This is done using the Silicon Laboratories IDE included with the Embedded Modem Development Kit.

1. To run the IDE, select **Silicon Laboratories IDE** from the **Start→Silicon Laboratories** menu or run *IDE.exe* located by default in the “C:\SiLabs\MCU” directory.
2. Select **Project→Open Project** to open the **Open Workspace** window. Browse to the project file created in Section 3.5 located in the “C:\SiLabs\MCU\Examples\Modem\Demo” directory. Select *TCPIP_Project.wsp*, and click the **Open** button.
3. Next, the file containing the Demo function must be added to the project. Right click on the **Source Files** file group in the **Project Workspace** window, and select **Add file to group Source Files**. Browse to the “C:\SiLabs\MCU\Examples\Modem\Demo” directory. Select *demo.c* and click the **Open** button.
4. In the **Project Workspace** window, right click on *demo.c*, and select **Add demo.c to build**.
5. Next, the `main()` function needs to be edited to call the function in *demo.c*. Open *main.c* and add `DEMO();` to the line above the final `while(1);` statement. After making the edit, Save and close *main.c*. The main function is listed below.

```
//-----
// Main Routine
//-----
void main (void) {

    PSOCKET_INFO socket_ptr;

    // Disable watchdog timer
    WDTCN = 0xde;
    WDTCN = 0xad;

    // Initialize the MCU
    PORT_Init ();
    SYSCLK_Init();

    // Initialize the TCP/IP stack
    if (mn_init() < 0)
    {
        // If code execution enters this while(1) loop, the stack failed to initialize.
        // Verify that all boards are connected and powered properly.
        while(1);
    }

    DEMO();

    while(1);

}
```

6. Next, the `DEMO()` function needs to be edited to include the correct connection information for the target modem. Open *demo.c* and edit the following lines using the **User name** and **Password** from Section 3.4, Step 8.

```
char pap_user[] = "user";
char pap_pwd[] = "password";
```

3.7. Running the Embedded Modem Demo

Once all the changes have been made to the firmware, the project must be built, downloaded to the C8051F120 device, and then run. Follow the instructions below to perform these steps.

1. Build the project by clicking on the **Build/Make Project** button in the toolbar or selecting **Project**→**Build/Make Project** from the menu.
2. Click the **Connect** button in the toolbar or select **Debug**→**Connect** from the menu to connect to the device.
3. Download the project to the target by clicking the **Download Code** button in the toolbar.
4. Run the demo by clicking the **Go** button in the toolbar or selecting **Debug**→**Go** from the menu.
5. To test the connection, the IP Address assigned to the Embedded Modem kit must be retrieved.
 - a. Open the **Network Connections** window from the **Control Panel**.
 - b. Wait for the connection to be established. When the connection is complete a Network status icon will appear in the system tray. Additionally, the connection, titled with the **User name**, will appear in the **Network Connections** window.
 - c. Double click on the **User name** connection to open the **Connection Status** window.
 - d. Select the **Details** tab and take note of the **Client IP Address** shown.
6. Open a Windows **Command Prompt** window to perform a ping test to verify that the Si2457 modem was successfully able to connect to the target modem
7. Using the IP Address from step 5, type the following in the **Command Prompt** window: ping *IP Address*
8. If the communication was successful "Reply" messages will appear. If the communication was NOT successful "Request Timed Out" messages will appear.

3.8. Troubleshooting

If the modems are unable to establish a connection, verify the following:

1. The `pap_user` and `pap_pwd` variables in *demo.c* correspond to the **User name** and **Password** of one of the users who is allowed to connect to the incoming connection.
2. It is possible that there is no DHCP server to allocate an IP address. From the **Network Connections** window, open up the connection and go to the **Networking** tab. Highlight **Internet Protocol (TCP/IP)** and click the **Properties** button. Under "TCP/IP address assignment," select "Specify TCP/IP addresses" and specify a range of IP addresses that is valid on your network in the "From:" and "To:" dialog boxes.

4. Software Setup

The included CD-ROM contains the Silicon Laboratories IDE, Keil software 8051 tools, and additional documentation. Insert the CD-ROM into your PC's CD-ROM drive. An installer will automatically launch, allowing you to install the IDE software or read documentation by clicking buttons on the Installation Panel. If the installer does not automatically start when you insert the CD-ROM, run *autorun.exe* found in the root directory of the CD-ROM. Refer to the *readme.txt* file on the CD-ROM for the latest information regarding known IDE problems and restrictions.

5. Silicon Laboratories IDE (Integrated Development Environment)

The Silicon Laboratories IDE integrates a source-code editor, source-level debugger and in-system Flash programmer. The use of third-party compilers and assemblers is also supported. This development kit includes the Keil Software A51 macro assembler, BL51 linker and evaluation version C51 'C' compiler. These tools can be used from within the Silicon Laboratories IDE.

5.1. System Requirements

The Silicon Laboratories IDE requirements:

- Pentium-class host PC running Microsoft Windows 95 or later, or Microsoft Windows NT or later.
- One available USB port.
- 64 MB RAM and 40 MB free HD space recommended.

5.2. Assembler and Linker

A full-version Keil A51 macro assembler and BL51 banking linker are included with the development kit and are installed during IDE installation. The complete assembler and linker reference manual can be found on-line under the **Help** menu in the IDE or in the "*SiLabs\MCU\hlp*" directory (A51.pdf).

5.3. Evaluation C51 'C' Compiler

An evaluation version of the Keil C51 'C' compiler is included with the development kit and is installed during IDE installation. The evaluation version of the C51 compiler is the same as the full professional version except code size is limited to 4 kB and the floating point library is not included. Additionally, support is included for importing TCP/IP Library object code. The C51 compiler reference manual can be found under the **Help** menu in the IDE or in the "*SiLabs\MCU\hlp*" directory (C51.pdf).

5.4. Using the Keil Software 8051 Tools with the Silicon Laboratories IDE

To perform source-level debugging with the IDE, you must configure the Keil 8051 tools to generate an absolute object file in the OMF-51 format with object extensions and debug records enabled. You may build the OMF-51 absolute object file by calling the Keil 8051 tools at the command line (e.g., batch file or make file) or by using the project manager built into the IDE. The default configuration when using the Silicon Laboratories IDE project manager enables object extension and debug record generation. Refer to application note "AN104: Integrating Keil 8051 Tools Into the Silicon Labs IDE" in the "*SiLabs\MCU\Documentation\Appnotes*" directory on the CD-ROM for additional information on using the Keil 8051 tools with the Silicon Laboratories IDE.

To build an absolute object file using the Silicon Laboratories IDE project manager, you must first create a project. A project consists of a set of files, IDE configuration, debug views, and a target build configuration (list of files and tool configurations used as input to the assembler, compiler, and linker when building an output object file).

The following sections illustrate the steps necessary to manually create a project with one or more source files, build a program and download the program to the target in preparation for debugging. (The IDE will automatically create a single-file project using the currently open and active source file if you select **Build/Make Project** before a project is defined.)

5.4.1. Creating a New Project

1. Select **Project→New Project** to open a new project and reset all configuration settings to default.
2. Select **File→New File** to open an editor window. Create your source file(s) and save the file(s) with a recognized extension, such as .c, .h, or .asm, to enable color syntax highlighting.
3. Right-click on “New Project” in the **Project Window**. Select **Add files to project**. Select files in the file browser and click Open. Continue adding files until all project files have been added.
4. For each of the files in the **Project Window** that you want assembled, compiled, and linked into the target build, right-click on the file name and select **Add file to build**. Each file will be assembled or compiled as appropriate (based on file extension) and linked into the build of the absolute object file.

Note: If a project contains a large number of files, the “Group” feature of the IDE can be used to organize. Right-click on “New Project” in the **Project Window**. Select **Add Groups to project**. Add pre-defined groups or add customized groups. Right-click on the group name and choose **Add file to group**. Select files to be added. Continue adding files until all project files have been added.

5.4.2. Building and Downloading the Program for Debugging

1. Once all source files have been added to the target build, build the project by clicking on the **Build/Make Project** button in the toolbar or selecting **Project→Build/Make Project** from the menu.

Note: After the project has been built the first time, the **Build/Make Project** command will only build the files that have been changed since the previous build. To rebuild all files and project dependencies, click on the **Rebuild All** button in the toolbar or select **Project→Rebuild All** from the menu.

2. Before connecting to the target device, several connection options may need to be set. Open the **Connection Options** window by selecting **Options→Connection Options...** in the IDE menu. First, select the appropriate adapter in the “Serial Adapter” section. Next, the correct “Debug Interface” must be selected. C8051F12x family devices use the JTAG debug interface. Once all the selections are made, click the **OK** button to close the window.
3. Click the **Connect** button in the toolbar or select **Debug→Connect** from the menu to connect to the device.
4. Download the project to the target by clicking the **Download Code** button in the toolbar.

Note: To enable automatic downloading if the program build is successful select **Enable automatic connect/download after build** in the **Project→Target Build Configuration** dialog. If errors occur during the build process, the IDE will not attempt the download.

5. Save the project when finished with the debug session to preserve the current target build configuration, editor settings and the location of all open debug views. To save the project, select **Project→Save Project As...** from the menu. Create a new name for the project and click on **Save**.

6. Example Source Code

Example source code, libraries, and register definition files are provided in the “*SiLabs\MCU\Examples\Modem*” directory during IDE installation. These files may be used as a template for code development. Example applications include an HTTP web server, SMTP mail client, TCP echo client/server, and UDP echo client/server. See “Appendix A—Protocols Supported by the TCP/IP Configuration Wizard” on page 24 for a short description of each protocol supported by the TCP/IP Configuration Wizard. Each of these examples were created using the TCP/IP Configuration Wizard.

6.1. HTTP Web Server

The example project *HTTP.wsp*, located by default in the “*SiLabs\MCU\Examples\Modem\HTTP*” directory, configures the embedded modem to act as an HTTP web server. The modem places an outgoing call and establishes a PPP connection with the PC. The number that the modem dials is defined as `MODEM_DIAL` in *mn_userconst.h*. The number must be in the format “ATDT#r” (where ‘#’ is the phone number of the PC’s modem with no spaces or dashes). The PC must be configured to accept incoming connections (See “Appendix B—Connecting the Embedded Modem to a PC” on page 25). The user can specify the modem’s IP address (`IP_SRC_ADDR` in *mn_userconst.h*) or the PC can be configured to assign an IP address to the modem. To establish a PPP connection, the variables `pap_user` and `pap_pwd` in *main.c* must match the username and password of one of the user profiles on the PC. Once the PPP connection has been established, the user can type “http://XXX.XXX.XXX.XXX” (where XXX.XXX.XXX.XXX is the IP address of the modem) into a web browser to view a “Hello World” HTML document served by the embedded modem.

Note: HTML files can be converted to C files using the *HTML2C.exe* utility located by default in the “*SiLabs\MCU\Utilities\HTML2C*” directory.

6.2. SMTP Mail Client

The example project *SMTP.wsp*, located by default in the “*SiLabs\MCU\Examples\Modem\SMTP*” directory, configures the embedded modem to act as an SMTP mail client. The modem places an outgoing call and establishes a PPP connection with the PC. The number that the modem dials is defined as `MODEM_DIAL` in *mn_userconst.h*. The number must be in the format “ATDT#r” (where ‘#’ is the phone number of the PC’s modem with no spaces or dashes). The PC must be configured to accept incoming connections (See “Appendix B—Connecting the Embedded Modem to a PC” on page 25). The user can specify the modem’s IP address (`IP_SRC_ADDR` in *mn_userconst.h*) or the PC can be configured to assign an IP address to the modem. To establish a PPP connection, the variables `pap_user` and `pap_pwd` in *main.c* must match the username and password of one of the user profiles on the PC. Once the PPP connection is established, the modem will send two emails to the SMTP server at the address defined by `IP_SMTP_ADDR` in *mn_userconst.h*. The SMTP server will then forward those emails to the email address defined by the variable `to[]` in *main.c*. One email has a text file attached, and the other has no attachments.

6.3. TCP Echo Client/Server

The example project *TCP.wsp*, located by default in the “*SiLabs\MCU\Examples\Modem\TCP\firmware*” directory, configures the embedded modem to act as either a TCP echo client or a TCP echo server. To place the modem in server mode, change the value of variable `SERVER_MODE` to 1 in *main.c*. To place it in client mode, change `SERVER_MODE` to 0. The user can specify the modem’s IP address (`IP_SRC_ADDR` in *mn_userconst.h*) or the PC can be configured to assign an IP address to the modem. To establish a PPP connection, the variables `pap_user` and `pap_pwd` in *main.c* must match the username and password of one of the user profiles on the PC.

In client mode, the modem places an outgoing call and establishes a PPP connection with the PC. The number that the modem dials is defined as `MODEM_DIAL` in *mn_userconst.h*. The number must be in the format "ATDT#r" (where '#' is the phone number of the PC's modem with no spaces or dashes). The PC must be configured to accept incoming connections (See "Appendix B—Connecting the Embedded Modem to a PC" on page 25). To run the example, first run the TCP Echo Server application ("*..\Modem\TCP\Application\TCP_SVR.exe*") on the PC and then start running code on the C8051F120. Once the PPP connection is established, TCP_SVR will wait for the embedded modem to send a string, then echo the received string back to the embedded modem. The embedded modem will then take the string from TCP_SVR and send it back and so forth.

In server mode, the modem waits for an incoming call from the PC before establishing a PPP connection. The PC must be configured to make an outgoing call to the modem (See "Appendix B—Connecting the Embedded Modem to a PC" on page 25). To run the example, first run code on the C8051F120 and then initiate a call from the PC. Once a PPP connection has been established, run the TCP Echo Client application ("*..\Modem\TCP\Application\TCP_CLI.exe*") on the PC. By default, TCP_CLI will send data to 216.233.5.26. If the embedded modem is using a different IP address, execute TCP_CLI from the command line using this syntax:

```
TCP_CLI XXX.XXX.XXX.XXX
```

where XXX.XXX.XXX.XXX is the IP address of the embedded modem.

TCP_CLI will send out a string and wait for the embedded modem to echo it back. TCP_CLI will then take the received string and send it to the embedded modem and so forth.

6.4. UDP Echo Client/Server

The example project *UDP.wsp*, located by default in the "*SiLabs\MCU\Examples\Modem\UDP\Firmware*" directory, configures the embedded modem to act as either a UDP echo client or a UDP echo server. To place the modem in server mode, change the value of variable `SERVER_MODE` to 1 in *main.c*. To place it in client mode, change `SERVER_MODE` to 0. The user can specify the modem's IP address (`IP_SRC_ADDR` in *mn_userconst.h*) or the PC can be configured to assign an IP address to the modem. To establish a PPP connection, the variables `pap_user` and `pap_pwd` in *main.c* must match the username and password of one of the user profiles on the PC.

In client mode, the modem places an outgoing call and establishes a PPP connection with the PC. The number that the modem dials is defined as `MODEM_DIAL` in *mn_userconst.h*. The number must be in the format "ATDT#r" (where '#' is the phone number of the PC's modem with no spaces or dashes). The PC must be configured to accept incoming connections (See "Appendix B—Connecting the Embedded Modem to a PC" on page 25). To run the example, first run the UDP Echo Server application ("*..\Modem\UDP\Application\UDP_SVR.exe*") on the PC and then start running code on the C8051F120. Once the PPP connection is established, UDP_SVR will wait for the embedded modem to send a string, then echo the received string back to the embedded modem. The embedded modem will then take the string from UDP_SVR and send it back and so forth.

In server mode, the modem waits for an incoming call from the PC before establishing a PPP connection. The PC must be configured to make an outgoing call to the modem (See "Appendix B—Connecting the Embedded Modem to a PC" on page 25). To run the example, first run code on the C8051F120 and then initiate a call from the PC. Once a PPP connection has been established, run the UDP Echo Client application ("*..\Modem\UDP\Application\UDP_CLI.exe*") on the PC. By default, UDP_CLI will send data to 216.233.5.26. If the embedded modem is using a different IP address, execute UDP_CLI from the command line using this syntax:

```
UDP_CLI XXX.XXX.XXX.XXX
```

where XXX.XXX.XXX.XXX is the IP address of the embedded modem.

UDP_CLI will send out a string and wait for the embedded modem to echo it back. UDP_CLI will then take the received string and send it to the embedded modem and so forth.

7. TCP/IP Configuration Wizard

The TCP/IP Configuration Wizard, shown in Figure 5, generates supporting directory structure and framework code required to start a new TCP/IP project. The wizard generates a custom library that describes the selected protocol configuration. To start using the code generated by the wizard, open the *TCPIP_Project.wsp* file using the Project→Open command from the Silicon Laboratories IDE. See application note “AN237: TCP/IP Library Programmer’s Guide” for detailed information about the TCP/IP stack.

Note: The CMX Micronet™ protocol stack is only licensed for use on Silicon Laboratories, Inc. MCU devices. Contact the Silicon Laboratories MCU support team at www.silabs.com for any questions or problems using the TCP/IP Library.

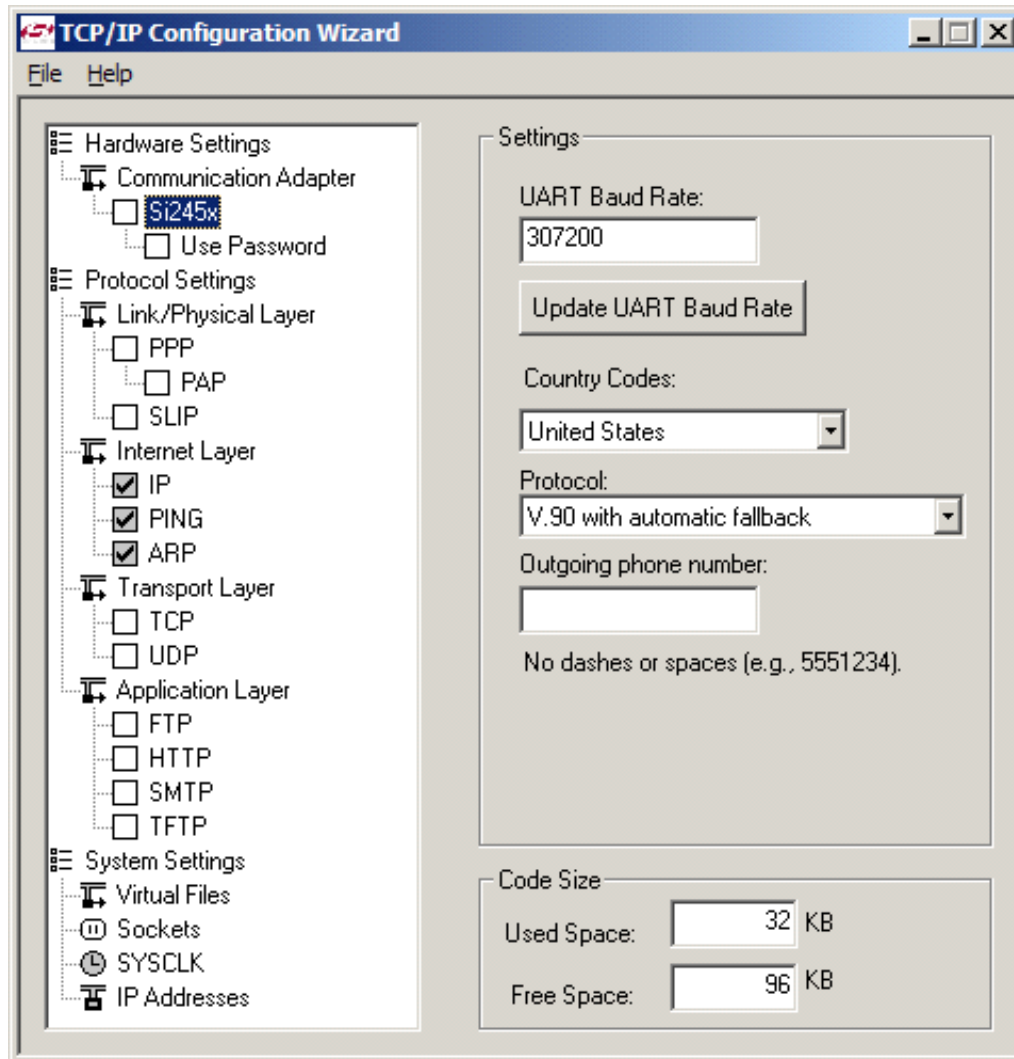


Figure 5. TCP/IP Configuration Wizard

8. C8051F120 Target Board

The Embedded Modem Development Kit includes a target board with a C8051F120 device pre-installed for evaluation and preliminary software development. Numerous input/output (I/O) connections are provided to facilitate prototyping using the target board. Refer to Figure 6 for the locations of the various I/O connectors. For further details, see the “C8051F12x Development Kit User’s Guide” located on the kit CD or installed by default in the “C:\SiLabs\MCU\Documentation\Users Guides” directory.

P1	Power connector (accepts input from 7 to 15 VDC unregulated power adapter)
J1	Connects SW2 to P3.7 pin
J3	Connects LED D3 to P1.6 pin
J4	JTAG connector for Debug Adapter interface
J5	DB-9 connector for UART0 RS232 interface
J6	Connector for UART0 TX (P0.0)
J8	Connector for UART0 RTS (P4.0)
J9	Connector for UART0 RX (P0.1)
J10	Connector for UART0 CTS (P4.1)
J11	Analog loopback connector
J12–J19	Port 0–7 connectors
J20	Analog I/O terminal block
J22	VREF connector
J23	VDD Monitor Disable
J24	96-pin Expansion I/O connector

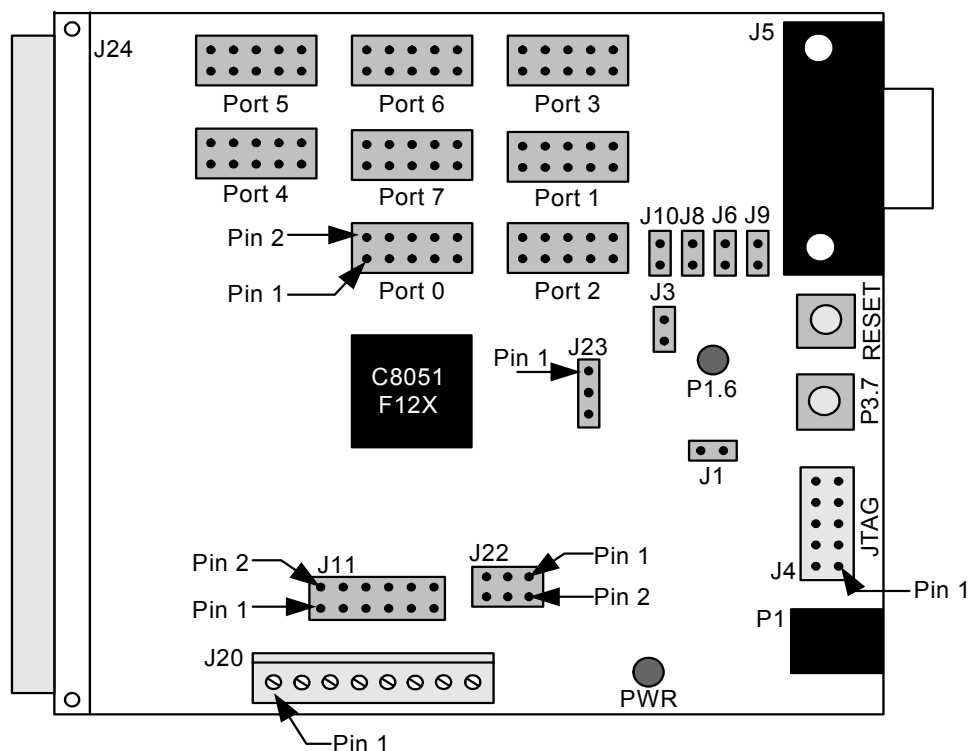


Figure 6. C8051F120 Target Board

9. AB3 Modem Adapter Board

The Embedded Modem Development Kit includes an adapter board designed to enable the C8051F120 device on the target board to communicate with the Si2457 modem device on the Si2457FT18-EVB Modem Board. This adapter board provides several LEDs and switches for use in modem communications. Refer to Figure 7 for the locations of the various I/O connectors.

J1	96-pin Expansion I/O connector
JP1	Connector for Si2457FT18-EVB Modem Board, modem signals
JP2	Connector for Si2457FT18-EVB Modem Board, phone line signals
JP4	PCM Interface connector
RJ11	Dual telephone jack

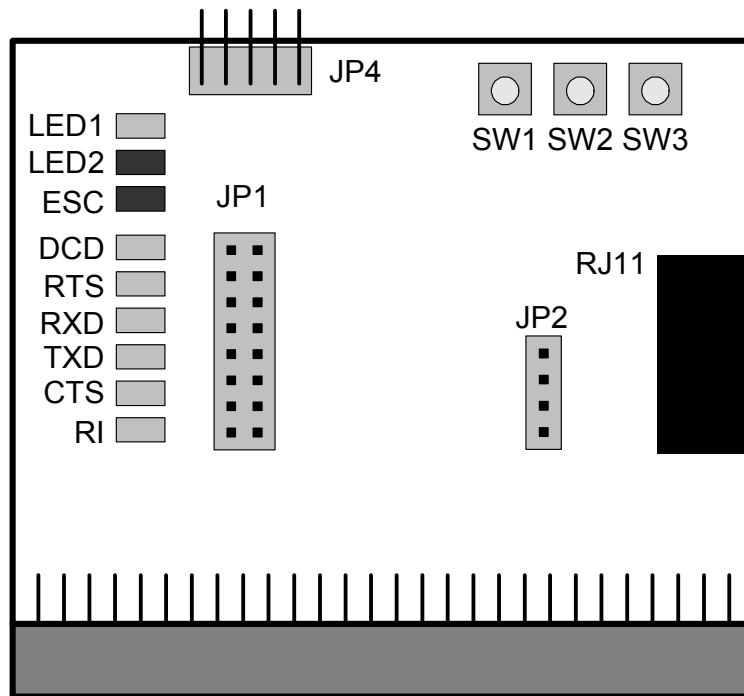


Figure 7. AB3 Modem Adapter Board

9.1. Switches and LEDs

Three switches are provided on the AB3 Modem Adapter Board. They are connected to C8051F120 GPIO (general purpose I/O) pins through the J1 96-pin connector. Pressing SW1, SW2, or SW3 generates a logic low signal on the port pin. See Table 1 for the port pins corresponding to each switch.

Nine LEDs are also provided on the adapter board. Two LEDs, one red and one green, are provided as general purpose indicators. These LEDs are connected to C8051F120 GPIO pins through the J1 96-pin connector. Seven LEDs, one red and six green, are provided as modem signal indicators. These LEDs are connected to C8051F120 GPIO pins that are used for modem signals to the Si2457 on the modem board. See Table 1 for the port pins corresponding to each LED.

Table 1. Target Board I/O Descriptions

Description	I/O
SW1	P3.4
SW2	P3.5
SW3	P3.6
LED1 (green)	P3.2
LED2 (red)	P3.3
ESC (red)	P3.1
DCD (green)	P1.1
RTS (green)	P2.7
RXD (green)	P0.7
TXD (green)	P0.6
CTS (green)	P1.0
RI (green)	P1.2

9.2. PCM Interface (JP4)

The Si2457 PCM interface is available on JP4. Table 2 lists the pin connections for JP4 designed to connect directly to the Si3000SSI-EVB JP6.

Table 2. JP4 Connector Pin Descriptions

JP4 Pin	AB3 Board Signal	Si2457 Pin	Si2457 Signal
1	CLKOUT	3	CLKOUT
2	TXCLK	4	FSYNC
3	GND	6, 20	GND
4	GND	6, 20	GND
5	RXCLK	24	SDO
6	EESD	18	SDI
7	RESET	12	RESET
8	3.3 V	5, 21	VD3.3
9	GND	6, 20	GND
10	3.3 V	5, 21	VD3.3

9.3. Modem Board Connectors (JP1, JP2)

The AB3 Modem Adapter Board motherboard connects to the Si2457FT18-EVB Modem Board through two connectors, JP1 and JP2. JP1 is an 8x2 socket providing connection to all Si2457 digital signals and regulated 3.3 V power for the Si2457. The Si2457 digital signals appearing at JP1 (modem board interface) are LVCMOS and TTL compatible. JP2 is a 4x1 socket providing connection between the modem board and the RJ11 phone jack. See Table 3 and Table 4 for JP1 and JP2 pin descriptions.

Table 3. JP2 Connector Pin Descriptions

Pin #	Description
1	TIP
2	RING
3, 4	NC

Table 4. JP1 Connector Pin Descriptions

JP1 Pin #	Si2457 Pin Descriptions		C8051F120 Pin Descriptions			
	Serial Interface	EEPROM Interface	PCA Interface	Serial Interface	Parallel Interface	EEPROM Interface
1	CLKOUT	EECS	P1.4 (CEX4)	P2.0 (SYSCLOCK)	P6.0 (A0)	P0.5 (NSS)
2	RXCLK	EECLK	P1.5 (CEX5)		P7.5 (Data Bus)	P0.2 (SCK)
3	TXCLK		P1.3 (CEX3)	P2.6 (TXCLK)	P7.6 (Data Bus)	
4	DCD		P1.1 (CEX1)		P7.4 (Data Bus)	
5	RTS			P2.7 (RTS)	P7.7 (Data Bus)	
6	GND					
7	RXD			P0.7 (RXD)	P4.6 (RD)	
8	ESC			P3.1 (GPO)	P7.3 (Data Bus)	
9	TXD			P0.6 (TXD)	P4.7 (WR)	
10		EESD			P7.2 (Data Bus)	P0.4 (MOSI), P0.3 (MISO)
11	CTS		P1.0 (CEX0)		P6.7 (CS)	
12	RI		P1.2 (CEX2)		P7.1 (Data Bus)	
13	RESET			P3.7 (Reset)		
14	INT			P1.7 (/INT1)	P7.0 (Data Bus)	
15	VD					
16	AOUT			AIN0.0 (AIN0.0), P3.0 (GPIO)	P1.6 (/INT0)	

9.4. Dual Telephone Jack (RJ11)

The RJ11 dual telephone jack connector provides an interface from the Si2457 modem to a phone line. Refer to Table 5 for pin definitions.

Table 5. RJ11 Connector Pin Descriptions

Pin #	Description
1, 2, 5, 6, 7, 8, 11, 12	NC
3, 9	TIP
4, 10	RING

9.5. Expansion I/O Connector (J1)

The 96-pin expansion I/O connector J1 is used to connect the AB3 Modem Adapter Board to the C8051F120 Target Board. J1 provides access to many C8051F120 signal pins. Pins for +3 V, digital ground, analog ground and the unregulated power supply (VUNREG) are also available. The VUNREG pin is connected directly to the unregulated +V pin of the P1 power connector. See Table 6 for a complete list of pins available at J1.

Table 6. J1 Pin Descriptions

Pin #	Description	Pin #	Description	Pin #	Description
A-1	+3 VD2 (+3.3 VDC)	B-1	DGND (Digital Gnd)	C-1	XTAL1
A-2	MONEN	B-2	P1.7	C-2	P1.6
A-3	P1.5	B-3	P1.4	C-3	P1.3
A-4	P1.2	B-4	P1.1	C-4	P1.0
A-5	P2.7	B-5	P2.6	C-5	P2.5
A-6	P2.4	B-6	P2.3	C-6	P2.2
A-7	P2.1	B-7	P2.0	C-7	P3.7
A-8	P3.6	B-8	P3.5	C-8	P3.4
A-9	P3.3	B-9	P3.2	C-9	P3.1
A-10	P3.0	B-10	P0.7	C-10	P0.6
A-11	P0.5	B-11	P0.4	C-11	P0.3
A-12	P0.2	B-12	P0.1	C-12	P0.0
A-13	P7.7	B-13	P7.6	C-13	P7.5
A-14	P7.4	B-14	P7.3	C-14	P7.2
A-15	P7.1	B-15	P7.0	C-15	P6.7
A-16	P6.6	B-16	P6.5	C-16	P6.4
A-17	P6.3	B-17	P6.2	C-17	P6.1
A-18	P6.0	B-18	P5.7	C-18	P5.6
A-19	P5.5	B-19	P5.4	C-19	P5.3
A-20	P5.2	B-20	P5.1	C-20	P5.0
A-21	P4.7	B-21	P4.6	C-21	P4.5
A-22	P4.4	B-22	P4.3	C-22	P4.2
A-23	P4.1	B-23	P4.0	C-23	TMS
A-24	TCK	B-24	TDI	C-24	TDO
A-25	/RST	B-25	DGND (Digital Gnd)	C-25	VUNREG
A-26	AGND (Analog Gnd)	B-26	DAC1	C-26	DAC0
A-27	CP1-	B-27	CP1+	C-27	CP0-
A-28	CP0+	B-28	VREF	C-28	VREFD
A-29	VREF0	B-29	VREF1	C-29	AIN0.7
A-30	AIN0.6	B-30	AIN0.5	C-30	AIN0.4
A-31	AIN0.3	B-31	AIN0.2	C-31	AIN0.1
A-32	AIN0.0	B-32	AGND (Analog Gnd)	C-32	AV+ (+3.3 VDC Analog)

10. Si2457FT18-EVB Modem Board

The Si2457FT18-EVB Modem Board includes a complete controller-based modem chipset with an integrated and programmable direct access arrangement (DAA) that meets global telephone line requirements. This board is a complete modem solution suited for use in an embedded system. The modem board requires a 3.3 V supply capable of providing at least 35 mA and communicates with the system via LVCMOS/TTL-compatible digital signals on JP1. When using the Embedded Modem Development Kit, the power supply and communication signals are provided by the C8051F120 Target Board through the AB3 Modem Adapter Board. The RJ-11 jack is connected via JP2. See Section 9.3 for JP1 and JP2 pin descriptions.

- | | |
|-----|---|
| JP1 | Connector for AB3 Modem Adapter Board, modem signals |
| JP2 | Connector for AB3 Modem Adapter Board, phone line signals |

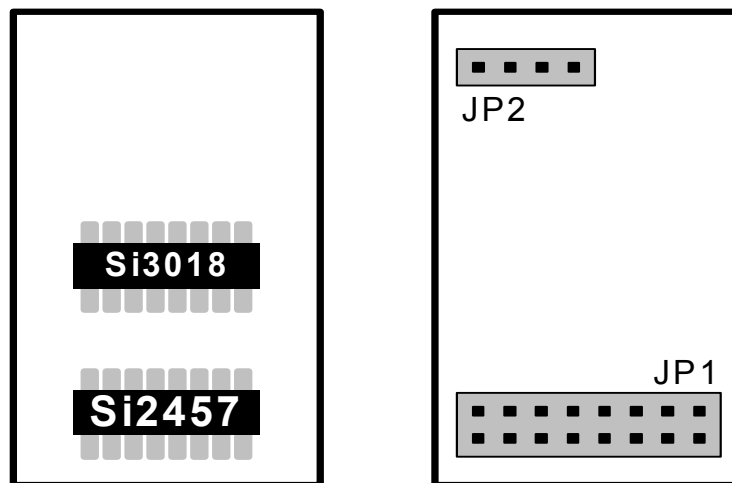


Figure 8. Si2457FT18-EVB Modem Board

11. Schematics

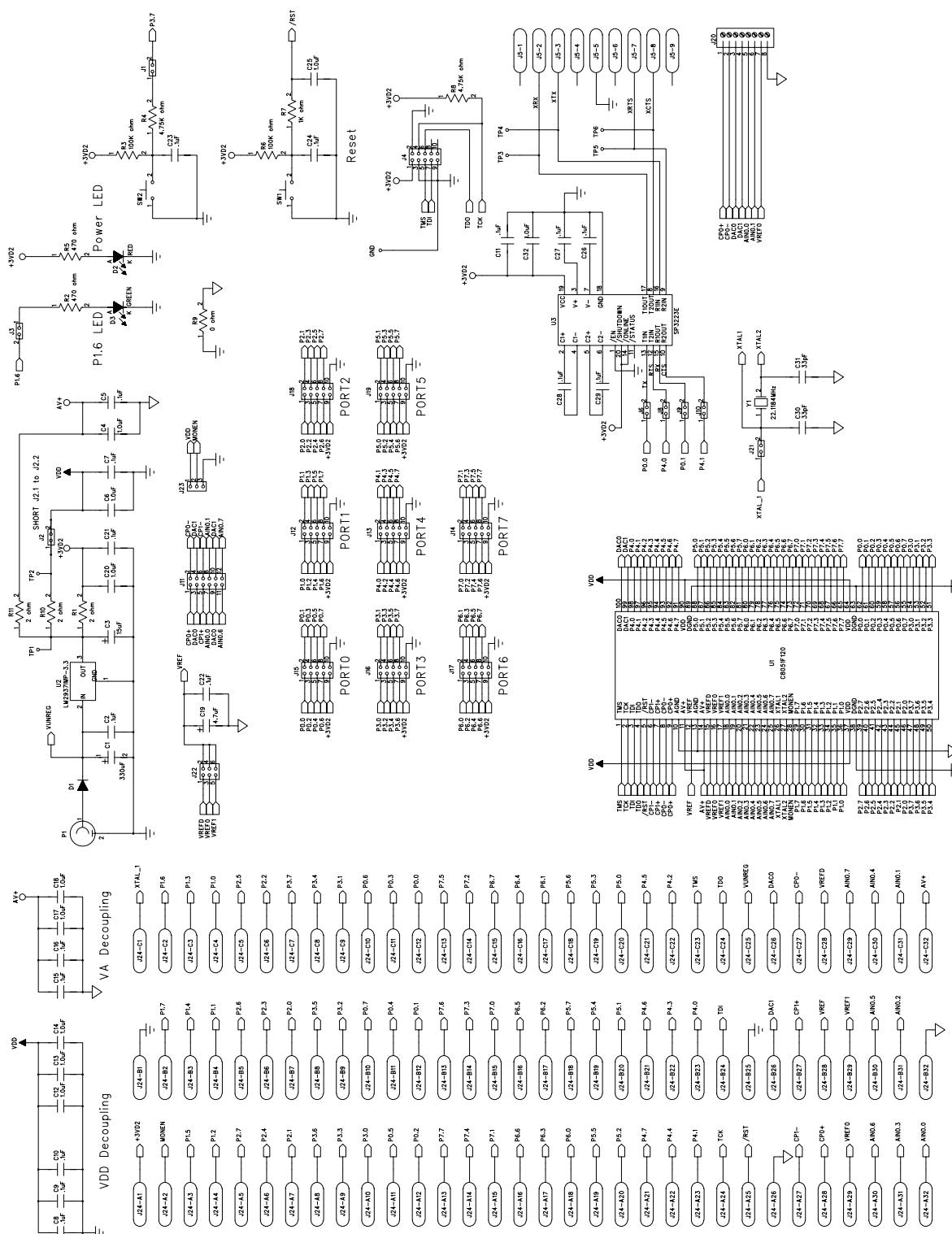


Figure 9. C8051F120 Target Board Schematic

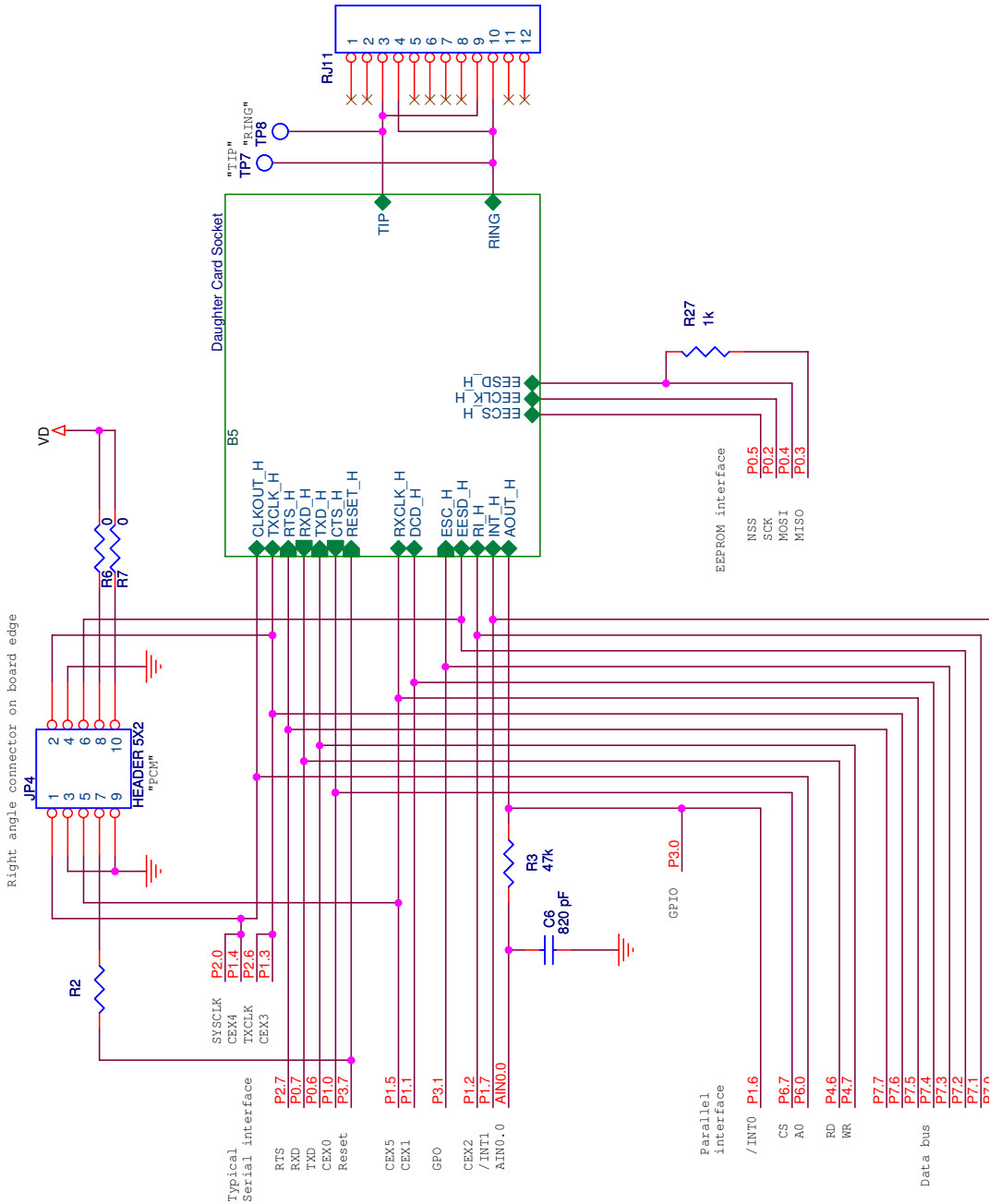


Figure 10. AB3 Modem Adapter Board Page 1

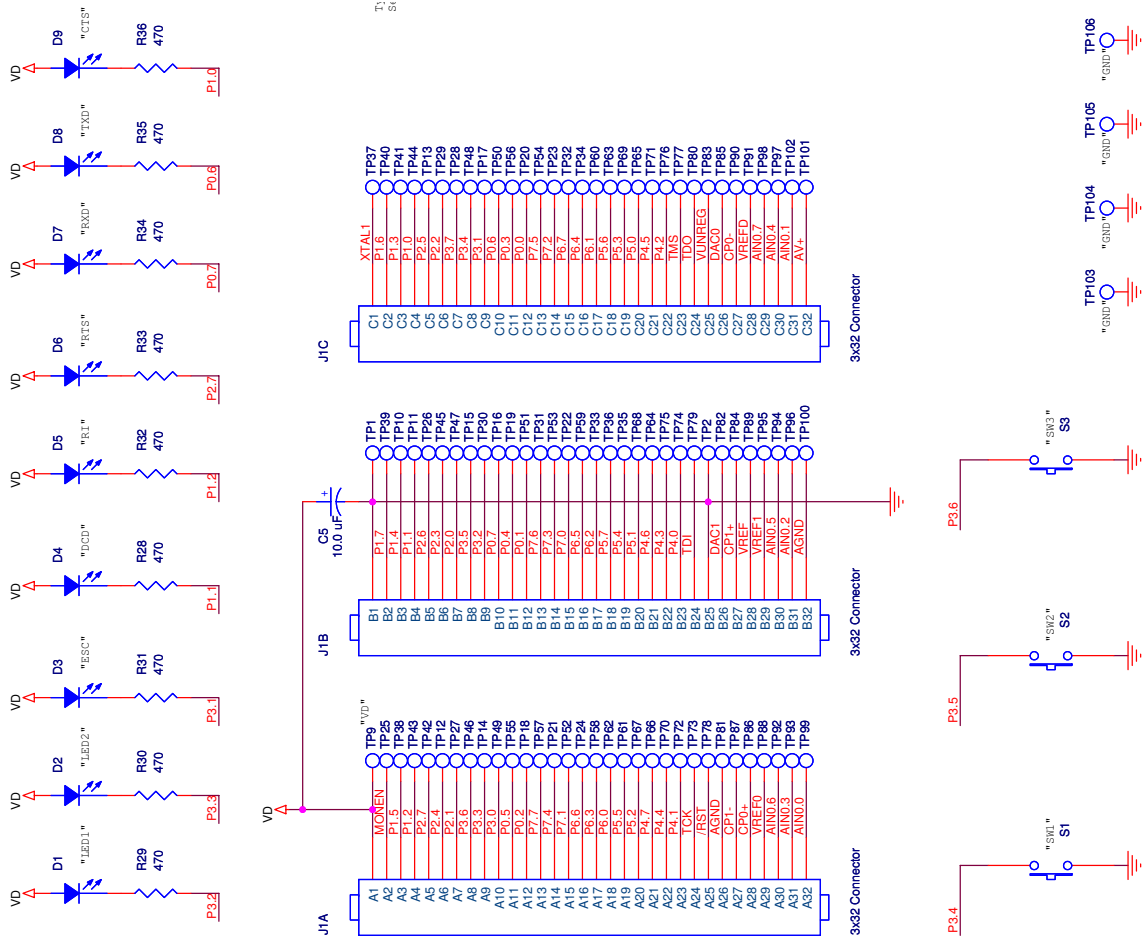
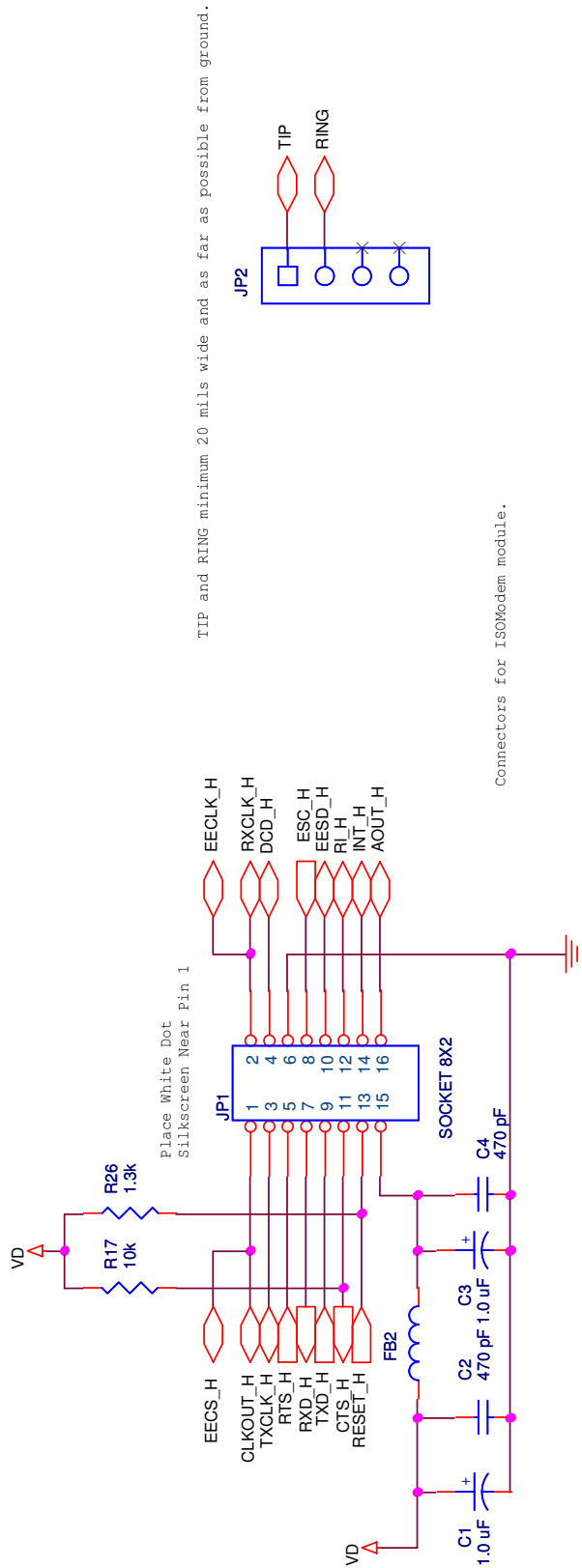


Figure 11. AB3 Modem Adapter Board Page 2



Net names correspond to Si24xx. See table for Si2400 equivalents

Table NOT on silkscreen

JP1/3	Si2400	Si24xx	Si2401
1	NC	CLKOUT/A0/EECS	NC
2	NC	EECLK/D5/RXCLK	GPIO1
3	NC	alt_R1b/TXCLK/D6	GPIO5
4	NC	DCD/D4	GPIO2
5	CLKOUT	RTS/D7	NC
6	GND	GND	GND
7	RXD	RXD/RD	RXD
8	GPIO1	ESC/D3	GPIO3
9	TXD	TXD/WR	TXD
10	GPIO2	EESD/D2	NC
11	CTS	CTS/CS	CTS
12	GPIO3	RI/D1	NC
13	RESET	RESET	RESET
14	GPIO4	INT/D0	NC
15	VD	VD	VD
16	AOUT	AOUT/INT	GPIO4

Figure 12. AB3 Modem Adapter Board Page 3

The schematic diagram illustrates the internal circuitry of the S2401 module. It features a central S2401 IC with various pins connected to external components. The 16-pin header (JP1) is connected to the IC pins, and the 24-pin header (JP2) is connected to the IC pins. The diagram includes a pin function table for the 16-pin header and a note about the 24-pin header.

Pin	S2401
1	GPIO0/RS0/TXCLK
2	GPIO1/RS1/TXCLK
3	GPIO2/RS2/TXCLK
4	GPIO3/RS3/TXCLK
5	GPIO4/RS4/TXCLK
6	GPIO5/RS5/TXCLK
7	GPIO6/RS6/TXCLK
8	GPIO7/RS7/TXCLK
9	GPIO8/RS8/TXCLK
10	GPIO9/RS9/TXCLK
11	GPIO10/RS10/TXCLK
12	GPIO11/RS11/TXCLK
13	GPIO12/RS12/TXCLK
14	GPIO13/RS13/TXCLK
15	GPIO14/RS14/TXCLK
16	GPIO15/RS15/TXCLK

These components are for internal use only. Silabs use only.

Figure 13. Si2457FT18-EVB Modem Board Page 1

No Ground Plane In DAA Section

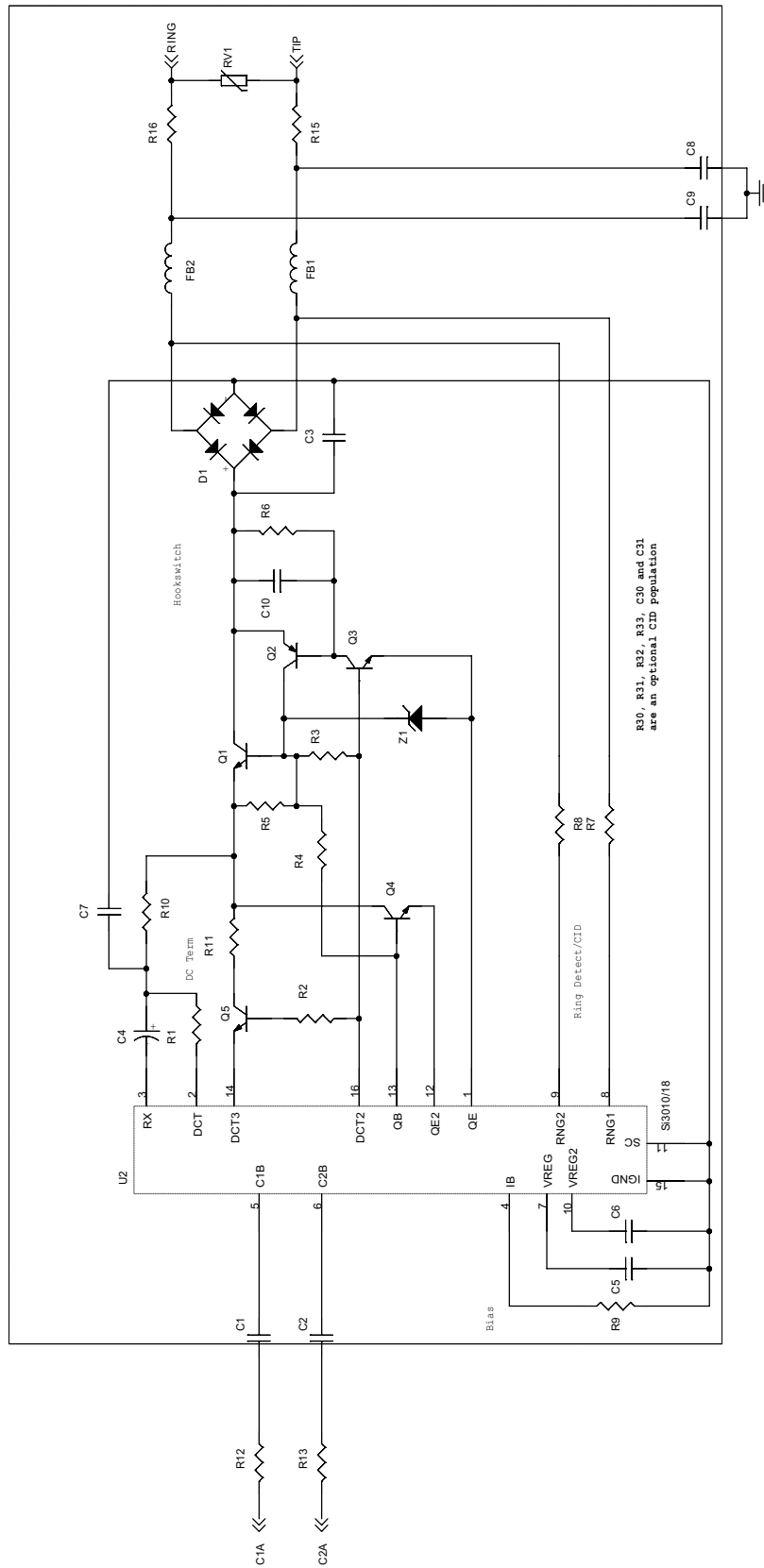


Figure 14. Si2457FT18-EVB Modem Board Page 2

APPENDIX A—PROTOCOLS SUPPORTED BY THE TCP/IP CONFIGURATION WIZARD

Link/Physical Layer:

SLIP (Serial Line Internet Protocol)—SLIP is a simple protocol for sending TCP/IP packages along a serial line. It provides no error-correction and requires that the device on each end of the connection know the identity of the device on the other end. SLIP is typically used on low noise RS232 links between two fixed processors.

PPP (Point to Point Protocol)—PPP is a more robust protocol for sending TCP/IP packages serially. PPP provides error correction and provides multiple options for configuring and connecting across a line. PPP is commonly used to send TCP/IP packages across modem lines.

PAP (Password Authentication Protocol)—PAP is a sub-protocol of PPP which requires that the device establishing the PPP connection supply a valid username and password.

Internet Layer:

IP (Internet Protocol)—IP specifies the format of packets and manages the addressing of all devices on the network. IP is included in all builds of the TCP/IP stack.

PING (Packet Internet Groper)—PING is a basic network program which verifies that a particular IP address exists and can respond to requests. PING is included in all builds of the TCP/IP stack.

ARP (Address Resolution Protocol)—ARP converts higher-level IP addresses to lower-level MAC addresses. It is not available with SLIP or PPP.

Transport Layer:

TCP (Transmission Control Protocol)—TCP is a connection-oriented transport protocol. It offers a reliable, full-duplex data stream that may be written to and read from by devices on the network.

UDP (User Datagram Protocol)—UDP is a connectionless, unreliable transport protocol. It formats data into units called datagrams and sends them across the network; however, it provides no guarantee that the data will arrive or that it will be error-free.

Application Layer:

FTP (File Transfer Protocol)—FTP transfers a complete file from one network device to another. FTP requires the TCP transport layer. The FTP module in the TCP/IP stack configures the device to operate as an FTP server only. It cannot act as an FTP client.

HTTP (HyperText Transfer Protocol)—HTTP sends and requests files (typically HTML files) across a network. It is commonly used to transfer files across the World Wide Web. HTTP requires the TCP transport layer. The HTTP module in the TCP/IP stack configures the device to operate as an HTTP server only. It cannot act as an HTTP client or browser.

SMTP (Simple Mail Transfer Protocol)—SMTP sends and receives e-mails. It requires the TCP transport layer. The SMTP module in the TCP/IP stack can only act as an SMTP client. An SMTP server must be available on the network to forward emails from the MODEM-DK to the specified e-mail address.

TFTP (Trivial File Transfer Protocol)—TFTP is a simplified version of FTP. It requires the UDP transport layer. The TFTP module in the TCP/IP stack can only act as a TFTP client. It cannot act as a TFTP server.

APPENDIX B—CONNECTING THE EMBEDDED MODEM TO A PC

The TCP/IP stack allows the embedded modem to be configured as a client or server. The embedded modem can communicate with any other modem through a standard telephone line (POTS) or telephone simulator. Any PC running Windows 2000 or Windows XP that has a modem can be configured to accept calls or dial into the embedded modem.

Configuring the PC to Accept Calls (Server Mode)

1. Go to the “Network Connections” dialog in the Control Panel.
2. Click on “Create New Connection”. The New Connection Wizard should appear.
3. If using Windows XP, select “Setup an Advanced Connection” and click “Next”.
4. Select “Accept Incoming Connections” and click “Next”.
5. Place a check mark next to the modem name.
6. Select “Do not allow virtual private connections”.
7. Place a check mark next to all user’s who will be allowed to use the modem.
8. Select Internet Protocol (TCP/IP) and click “Properties”. From this dialog, you can specify the IP address configuration and provide or restrict access to the local area network (LAN). Providing the embedded modem access to the LAN allows the embedded system to send e-mail using the SMTP mail server on a corporate network.
9. Click “Finish” to complete the connection.

Configuring the PC to Dial the Embedded Modem (Client Mode)

1. Go to the “Network Connections” dialog in the Control Panel.
2. Click on “Create new connection”. The New Connection Wizard should appear.
3. Select “Connect to the Internet” and click “Next”.
4. Select “Setup my connection manually” and click “Next”.
5. Select “Connect using a dial-up modem” and click “Next”.
6. Specify a name for the connection and click “Next”.
7. Specify the phone number of the embedded modem.
8. Select the user’s allowed to dial the connection.
9. Specify the user name and password used to log into the embedded modem.
10. Click “Finish” to complete the connection.

DOCUMENT CHANGE LIST

Revision 0.1 to Revision 0.2

- Removed Section 9. USB Debug Adapter. See USB Debug Adapter User's Guide.

NOTES:

CONTACT INFORMATION

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032
Email: MCUinfo@silabs.com
Internet: www.silabs.com

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories and Silicon Labs are trademarks of Silicon Laboratories Inc.

Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.