PRODUCT MANUAL

802.11b (Wi-Fi®) Add-On Kits

for RabbitCore and PowerCore Modules

User's Manual

019–0092 • 070831–F

802.11b (Wi-Fi®) Add-On Kits User's Manual

Part Number 019-0092 • 070831–F • Printed in U.S.A.

Trademarks

Rabbit and Dynamic C are registered trademarks of Rabbit Semiconductor Inc.

Rabbit 3000 and RabbitCore are trademarks of Rabbit Semiconductor Inc.

Wi-Fi and the Wi-Fi logo are registered trademarks of the Wi-Fi Alliance.

The latest revision of this manual is available on the Rabbit Semiconductor Web site, www.rabbit.com, for free, unregistered download.

Rabbit Semiconductor Inc.

www.rabbit.com

TABLE OF CONTENTS

1. INTRODUCTION

Rabbit Semiconductor's Wi-Fi Add-On Kits allow embedded system integrators to add Wi-Fi (IEEE 802.11b) to an existing embedded control system that uses a RabbitCore or PowerCore module based on the Rabbit® 3000 microprocessor. Wireless connectivity eliminates Ethernet cables, allowing for the greater flexibility and mobility associated with wireless embedded networks.

The Wi-Fi Add-On Kits provide an Interposer Board tailored to the footprint of your existing RabbitCore or PowerCore module, a CompactFlash Wi-Fi Board, a ribbon cable to connect the Interposer Board to the CompactFlash Wi-Fi Board, a Wi-Fi CompactFlash card, assorted mounting hardware, and two CD-ROMs, one with a Dynamic C® upgrade, the other with sample programs, software, and documentation related to the Wi-Fi Add-On Kits.

The Interposer Board can be inserted between the RabbitCore or PowerCore module and its Prototyping Board to allow you to use the sample programs and to eventually develop your own application. You can then use the Interposer Board with your existing motherboard to implement the Wi-Fi application you have developed, or you can add Wi-Fi capability to your motherboard based on the Interposer Board design.

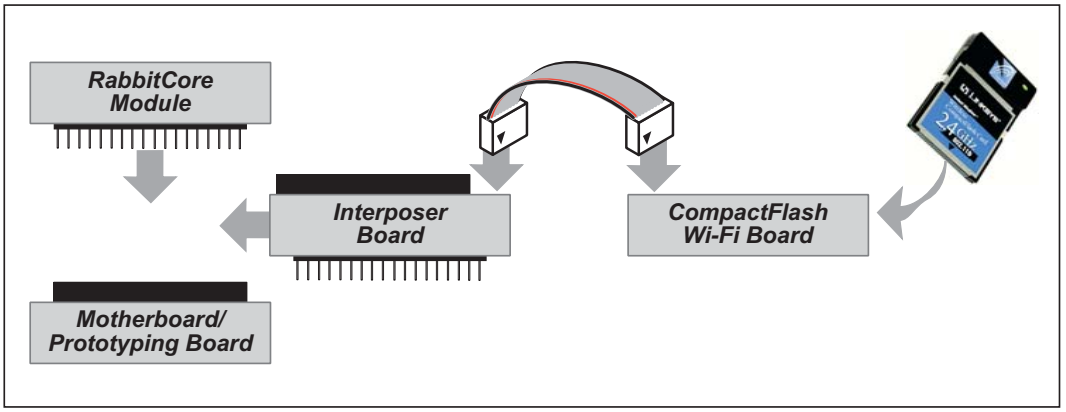


Figure 1. Using an Interposer Board to Add Wi-Fi to an Existing Embedded Control System

The CompactFlash Wi-Fi Board may be located up to 3 ft (90 cm) from your embedded control system, depending on bus loading and operating conditions, and can be chassis-mounted in a way that the Wi-Fi CompactFlash card can be removed or replaced.

1.1 Wi-Fi Add-On Kits

Four Wi-Fi Add-On Kits are available with Interposer Boards designed for the four different footprints of RabbitCore or PowerCore modules based on the Rabbit 3000 microprocessor. Table 1 below lists the four Wi-Fi Add-On Kits.

Table 1. Wi-Fi Add-On Kits

Interposer Board Footprint	Used with RabbitCore or PowerCore Modules	Used with Prototyping Board
RCM3000–RCM3300	RCM3000 series RCM3100 series RCM3200 series	RCM3000 Prototyping Board
	RCM3300 series	RCM3300 Prototyping Board
RCM3400	RCM3400 series	RCM3400 Prototyping Board
RCM3600–RCM3700	RCM3600 series RCM3700 series	RCM3600–RCM3700 Prototyping Board
PowerCore	PowerCore FLEX	PowerCore Prototyping Board

Appendix A provides detailed specifications for the Interposer Boards and the CompactFlash Wi-Fi Board.

1.2 Wi-Fi Applications in Embedded Control Systems

- Wireless automation and control
- Convenient data and event logging
- Instrumentation monitoring

1.3 Software

All Wi-Fi development involving the Wi-Fi Add-On Kits is done using version 9.21 or later of Rabbit Semiconductor's Dynamic C. A Dynamic C upgrade CD is included with the Wi-Fi Add-On Kits to allow you to upgrade your existing Dynamic C installation prior to installing the software from the supplementary CD-ROM in the Add-On Kits that contains the Wi-Fi sample programs and drivers.

Rabbit Semiconductor also offers for purchase add-on Dynamic C modules including the popular µC/OS-II real-time operating system, as well as point-to-point protocol (PPP), Advanced Encryption Standard (AES), FAT file system, Secure Sockets Layer (SSL), RabbitWeb, and other select libraries. In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase. Visit our Web site at www.rabbit.com for further information and complete documentation for each module, or contact your Rabbit Semiconductor sales representative or authorized distributor.

1.4 Online Documentation

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, use your browser to find and load **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

A separate icon will also appear on your workstation's desktop for the Wi-Fi documentation when you install the software from the supplementary CD-ROM in the Add-On Kits.

The latest versions of all documents are always available for free, unregistered download from our Web sites as well.

2. GETTING STARTED

This chapter explains how to set up and use a RabbitCore or PowerCore module based on a Rabbit 3000 microprocessor with an Interposer Board from the corresponding Wi-Fi Add-On Kit.

NOTE: To use a Wi-Fi Add-On Kit, you must already have a RabbitCore or PowerCore module based on the Rabbit 3000 microprocessor, the Prototyping Board for that RabbitCore or PowerCore module, a programming cable, and Dynamic C. All of these components are included in the Development Kit associated with your RabbitCore or PowerCore module.

2.1 Install Dynamic C

To develop and debug programs for RabbitCore or PowerCore modules (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C.

If you have not yet installed Dynamic C, do so now by inserting the Dynamic C CD from the Development Kit in your PC's CD-ROM drive. If autorun is enabled, the CD installation will begin automatically. If autorun is disabled or the installation otherwise does not start, use the Windows **Start > Run** menu or Windows Disk Explorer to launch **`setup.exe`** from the root folder of the CD-ROM.

The installation program will guide you through the installation process. Most steps of the process are self-explanatory.

In order to run the sample programs and use the Wi-Fi drivers from a Wi-Fi Add-On Kit, you must have Dynamic C 9.21 or a later version installed. If your version of Dynamic C is earlier than 9.21, install the Dynamic C upgrade included with your Wi-Fi Add-On Kit.

If you have purchased any of the optional Dynamic C modules, install them after installing Dynamic C and the Dynamic C upgrade. The modules may be installed in any order. You must install the modules in the same directory where Dynamic C was installed.

Finally, install the Wi-Fi software and sample programs from the supplementary CD included with your Wi-Fi Add-On Kit.

Once your installation is complete, you will have several icons on your PC desktop. One icon is for Dynamic C, one opens the Rabbit Field Utility, a tool used to download precompiled software to a target system, and two icons allow access to the documentation.

2.2 Hardware Connections

There are four steps to connecting the Prototyping Board for use with Dynamic C and the sample programs:

1. Insert the Interposer Board between the RabbitCore or PowerCore module and the Prototyping Board.
2. Use the ribbon cable from the Wi-Fi Add-On Kit to connect the Interposer Board to the CompactFlash Wi-Fi Board with a Wi-Fi CompactFlash card installed.
3. Connect the programming cable between the RabbitCore or PowerCore module and the workstation PC.
4. Connect the power supply to the Prototyping Board.

> **NOTE:** Make sure that no power is applied whenever you assemble or disassemble the array consisting of a RabbitCore/PowerCore module, Interposer Board, and Prototyping Board.

These steps are described in the following sections for the four Wi-Fi Add-On Kits. A printed copy of the connection instructions is included with each Wi-Fi Add-On Kit.

2.2.1 RCM3000–RCM3300 Wi-Fi Add-On Kit

1. Install RabbitCore Module and Interposer Board on Prototyping Board

Line up the pins on the bottom of the RCM3000–RCM3300 Interposer Board with sockets RCM2JA and RCM2JB on the RCM30/31/32XX Prototyping Board or with sockets JA and JB on the RCM3300 Prototyping Board (these have the same relative positions as sockets RCM2JA and RCM2JB on the RCM30/31/32XX Prototyping Board). Press the Interposer Board's pins firmly into the Prototyping Board sockets.

Next, line up the pins on the bottom of your RabbitCore module with sockets JA and JB on the Interposer Board. Press the module's pins firmly into the Interposer Board sockets.

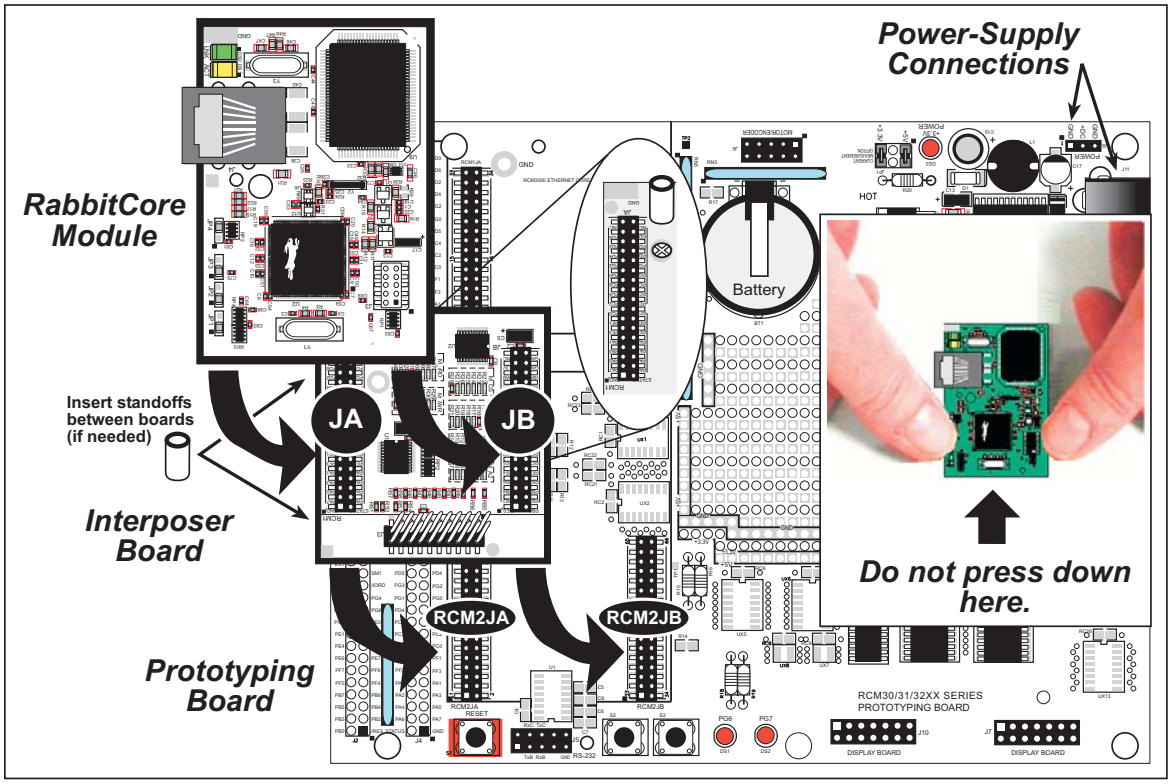


Figure 2. Install the RabbitCore Module and Interposer Board on the Prototyping Board

NOTE: It is important that you line up the pins on the RabbitCore module and the Interposer Board exactly with the corresponding pins of the sockets—press down in the area above the header pins using your thumbs or fingers over the connectors as shown in Figure 2. The header pins may become bent or damaged if the pin alignment is offset, and the boards will not work. Permanent electrical damage to the modules may also result if a misaligned module is powered up.

Two standoffs are included in the Add-On Kit to secure the RabbitCore module and the Interposer Board to the RCM3000 Prototyping Board in high-vibration environments. Attach one standoff to each side of the Interposer Board with the 2-56 × 3/16" screws included in the Add-On Kit as shown in Figure 2. Then place a 2-56 × 3/16" screw from the bottom side of the RCM3000 Prototyping Board into one standoff to secure the Interposer Board to the RCM30/31/32XX Prototyping Board, and add a 2-56 × 3/16" screw from the top side of the RabbitCore module into the other standoff on the Interposer Board to hold the RabbitCore module firmly in place.

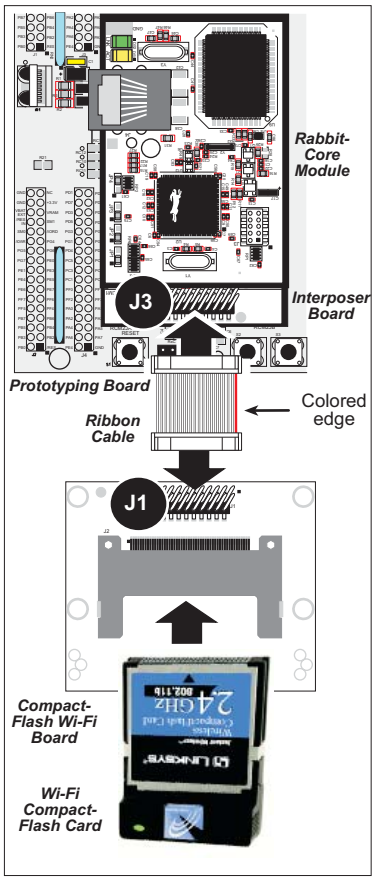


Figure 3. Connect Compact-Flash Wi-Fi Board

2. Connect CompactFlash Wi-Fi Board

Now orient the CompactFlash Wi-Fi Board as shown in Figure 3, and use the 20-pin ribbon cable supplied with the Add-On Kit to connect header J1 of the CompactFlash Wi-Fi Board to header J3 of the Interposer Board. Line up the colored (usually red) edge of the cable towards pin 1 of the two headers. Then insert the Wi-Fi CompactFlash card into the CompactFlash Wi-Fi Board as shown.

3. Connect Programming Cable

Connect the 10-pin connector of the programming cable labeled **PROG** to the programming header on your RabbitCore module according to the ***Getting Started*** instructions for your specific RabbitCore module. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the header. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

NOTE: Be sure to use the programming cable supplied with your RabbitCore module. Programming cables from other Rabbit Semiconductor kits might not be compatible with your RabbitCore module.

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 540-0070) with the programming cable supplied with the Tool Kit. Note that not all RS-232/USB converters work with Dynamic C.

4. Connect Power Supply

Finally, connect the wall transformer to jack J11 on the Prototyping Board as shown in Figure 4. The power LED on the Prototyping Board should light up.

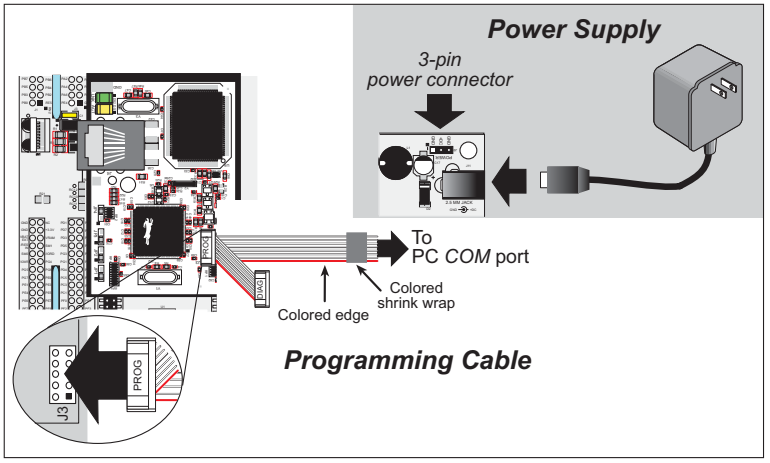


Figure 4. Connect Programming Cable and Power Supply

NOTE: The **RESET** button is provided on the Prototyping Board to allow a hardware reset without disconnecting power.

2.2.2 RCM3400 Wi-Fi Add-On Kit

1. Install RCM3400 Module and Interposer Board on Prototyping Board

Attach the plastic shock-absorber bumper to the bottom side of the Interposer Board under the 20-pin header as shown in Figure 5. Turn the RCM3400 Interposer Board so that the header with pins is on the left as shown in Figure 5 below. Align the Interposer Board's pins from headers J1 and J2 on the bottom side of the Interposer Board into sockets RCM1JA and RCM1JB on the Prototyping Board. The shaded corner notch at the top left corner of the Interposer Board should face the same direction as the corresponding notch below it on the Prototyping Board. Press the Interposer Board's pins firmly into the Prototyping Board sockets.

Turn the RCM3400 module so that the Rabbit 3000 chip is facing up and the Rabbit logo is facing the direction shown in Figure 5 below. Align the RCM3400 module's pins from headers J1 and J2 on the bottom side of the module into sockets RCM1JA and RCM1JB on the Interposer Board. The shaded corner notch at the top left corner of the RCM3400 module should face the same direction as the corresponding notch below it on the Interposer Board. Press the module's pins firmly into the Interposer Board sockets.

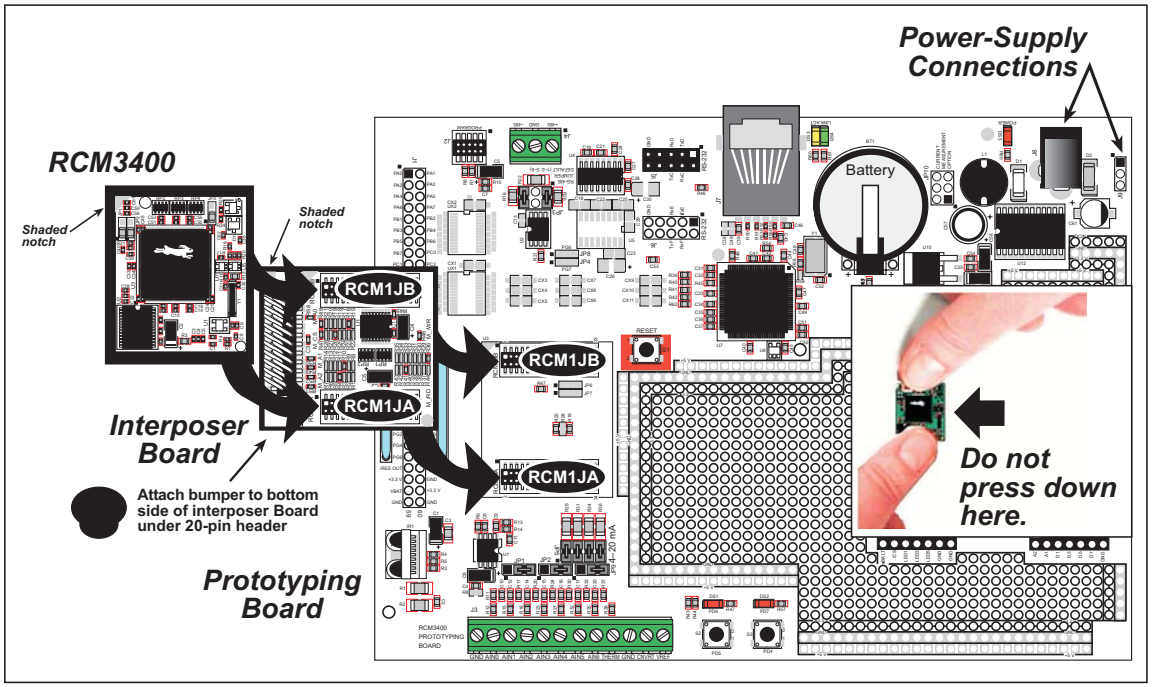


Figure 5. Install the RCM3400 Module and the Interposer Board on the Prototyping Board

NOTE: It is important that you line up the pins on the RCM3400 module and the Interposer Board exactly with the corresponding pins of the sockets—press down in the area above the header pins using your thumbs or fingers over the connectors as shown in Figure 5. The header pins may become bent or damaged if the pin alignment is offset, and the boards will not work. Permanent electrical damage to the module may also result if a misaligned module is powered up. Gently rock a board if you need to remove it so as not to damage the header and socket solder connections on the board.

2. Connect CompactFlash Wi-Fi Board

Now orient the CompactFlash Wi-Fi Board as shown in Figure 6, and use the 20-pin ribbon cable supplied with the Add-On Kit to connect header J1 of the CompactFlash Wi-Fi Board to header J3 of the Interposer Board. Line up the colored (usually red) edge of the cable towards pin 1 of the two headers. Then insert the Wi-FI CompactFlash card into the CompactFlash Wi-Fi Board as shown.

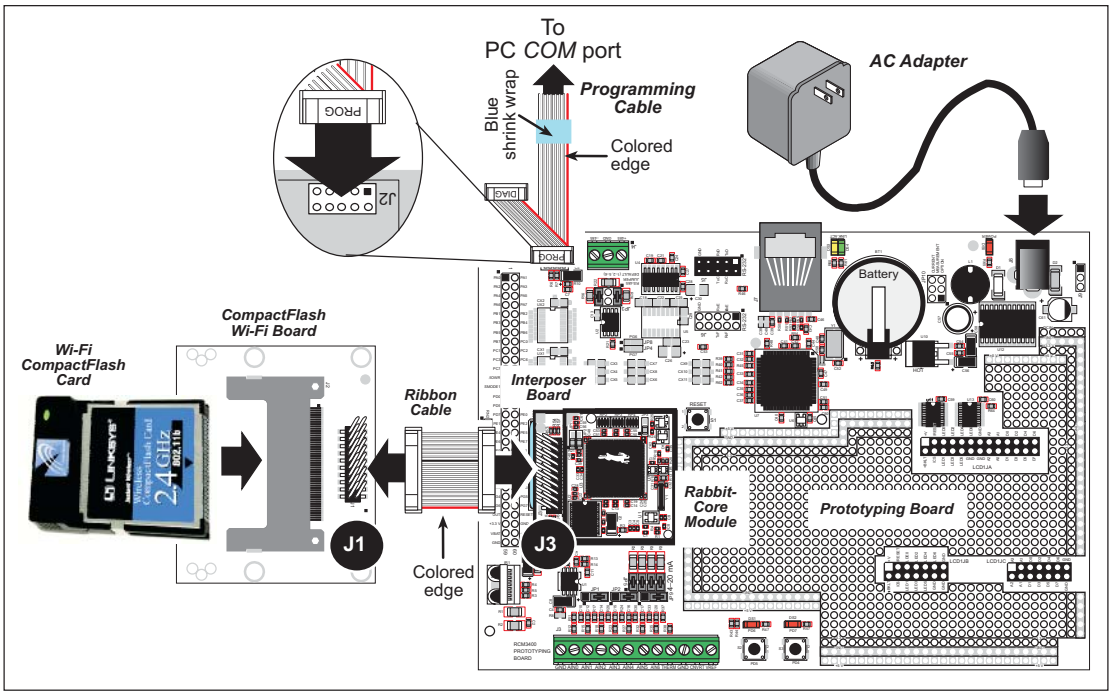


Figure 6. Connect CompactFlash, Programming Cable, and Power Supply

3. Connect Programming Cable

Connect the 10-pin connector of the programming cable labeled **PROG** to header J2 on the RCM3400 Prototyping Board as shown in Figure 6. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the header. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

NOTE: Be sure to use the programming cable supplied with your RabbitCore module—the programming cable has blue shrink wrap around the RS-232 converter section located in the middle of the cable. Programming cables from other Rabbit Semiconductor kits might not be compatible with your RabbitCore module.

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 540-0070) with the programming cable supplied with the Tool Kit. Note that not all RS-232/USB converters work with Dynamic C.

4. Connect Power Supply

Connect the wall transformer to jack J8 on the Prototyping Board as shown in Figure 6. Plug in the wall transformer. The power LED on the Prototyping Board should light up. The RCM3400 and the Prototyping Board are now ready to be used.

NOTE: The **RESET** button is provided on the Prototyping Board to allow a hardware reset without disconnecting power.

2.2.3 RCM3600–RCM3700 Wi-Fi Add-On Kit

1. Install RabbitCore module and Interposer Board on Prototyping Board

Turn the RCM3600–RCM3700 series Interposer Board so that the header with pins is on the right as shown in Figure 7 below. Press the Interposer Board's pins on the bottom side of the Interposer Board into the TCM_SMT_SOCKET socket on the Prototyping Board. The shaded corner at the bottom right corner of the Interposer Board should face the same direction as the corresponding shaded corner below it on the Prototyping Board.

Turn your RabbitCore module so that the shaded corner at the bottom right corner is on the bottom right as shown in Figure 7 below. Press the RabbitCore module's pins on the bottom side of the module firmly into the Interposer Board's RCM1 socket.

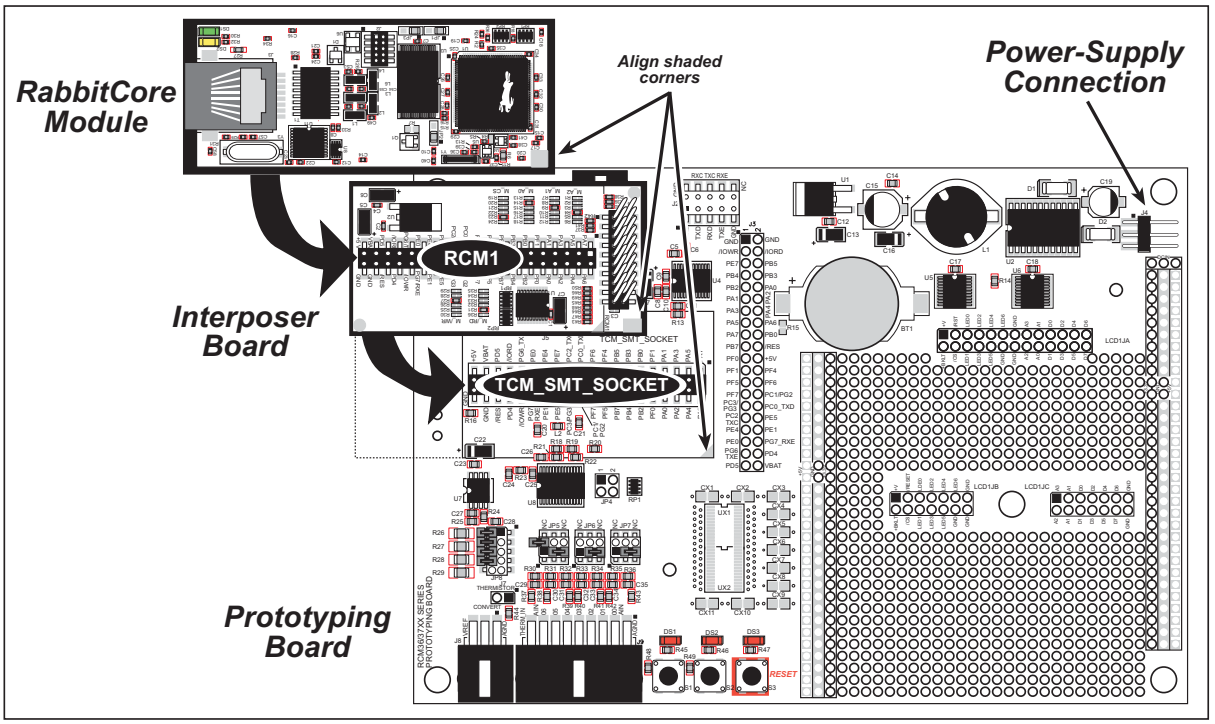


Figure 7. Install the RabbitCore Module and Interposer Board on the Prototyping Board

NOTE: It is important that you line up the pins on the RabbitCore module and the Interposer Board exactly with the corresponding pins of the sockets. The header pins may become bent or damaged if the pin alignment is offset, and the boards will not work. Permanent electrical damage to the modules may also result if a misaligned module is powered up.

2. Connect CompactFlash Wi-Fi Board

Now orient the CompactFlash Wi-Fi Board as shown in Figure 8, and use the 20-pin ribbon cable supplied with the Add-On Kit to connect header J1 of the CompactFlash Wi-Fi Board to header J2 of the Interposer Board. Line up the colored (usually red) edge of the cable towards pin 1 of the two headers. Then insert the Wi-Fi CompactFlash card into the CompactFlash Wi-Fi Board as shown.

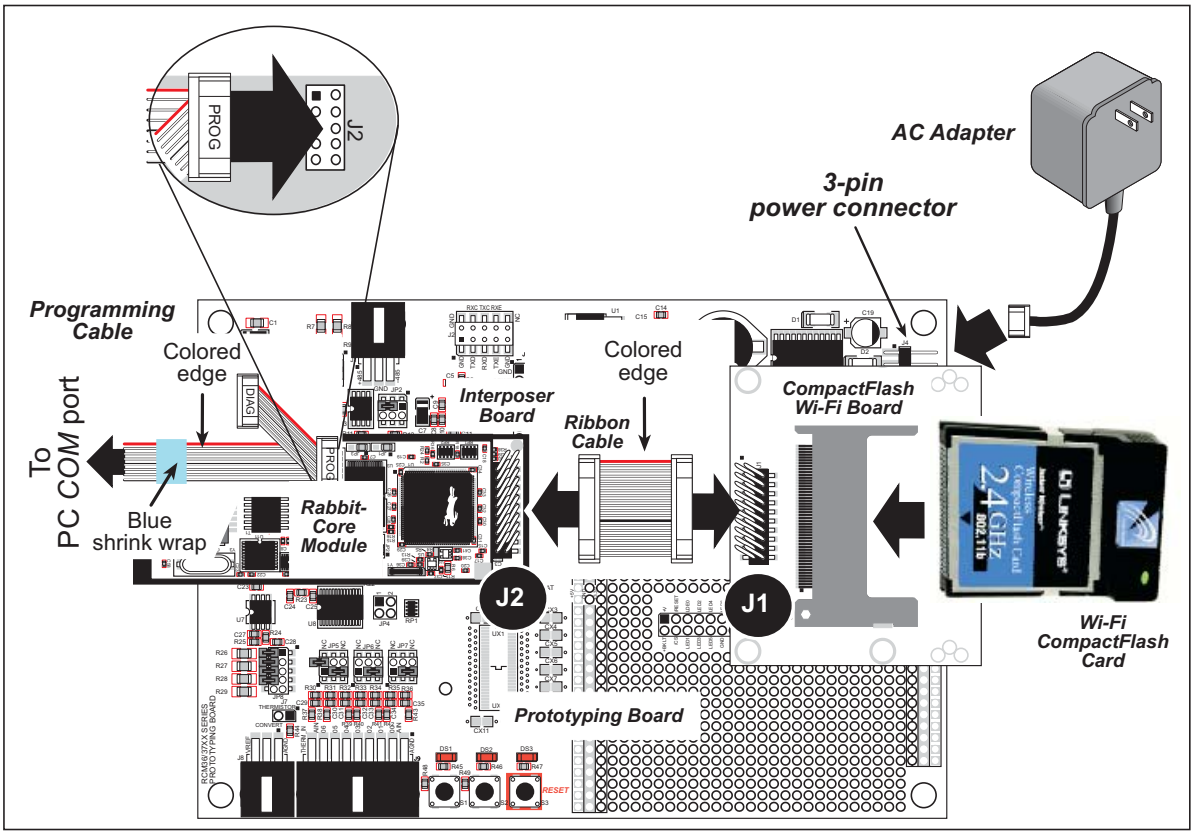


Figure 8. Connect CompactFlash, Programming Cable, and Power Supply

3. Connect Programming Cable

Connect the 10-pin connector of the programming cable labeled **PROG** to the programming header on your RabbitCore module according to the ***Getting Started*** instructions for your specific RabbitCore module. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the header. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

NOTE: Be sure to use the programming cable supplied with your RabbitCore module—the programming cable has blue shrink wrap around the RS-232 converter section located in the middle of the cable. Programming cables from other Rabbit Semiconductor kits might not be compatible with your RabbitCore module.

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 540-0070) with the programming cable supplied with the Tool Kit. Note that not all RS-232/USB converters work with Dynamic C.

4. Connect Power Supply

Connect the wall transformer to 3-pin header J4 on the Prototyping Board as shown in Figure 8. The connector may be attached either way as long as it is not offset to one side. The power LED on the Prototyping Board should light up.

NOTE: The **RESET** button is provided on the Prototyping Board to allow a hardware reset without disconnecting power.

2.2.4 PowerCore Wi-Fi Add-On Kit

1. Install PowerCore module and Wi-Fi Interposer Board on Prototyping Board

Snap in at least one standoff to each side of the Interposer Board as shown in Figure 9. Turn the Interposer Board so that the header with pins is on the bottom as shown in Figure 9 below. Press the Interposer Board's pins on the bottom side of the Interposer Board into the PC1 socket on the Prototyping Board. Press the Interposer Board's pins firmly into the Prototyping Board sockets.

NOTE: If your PowerCore Prototyping Board is at Rev. C or earlier, you will have to solder a 100 kΩ resistor across the pads at R49 on the Interposer Board.

Next, line up the header J4 pins on the bottom side of your PowerCore module with the PC1 socket on the Interposer Board. Press the module's pins firmly into the Interposer Board socket.

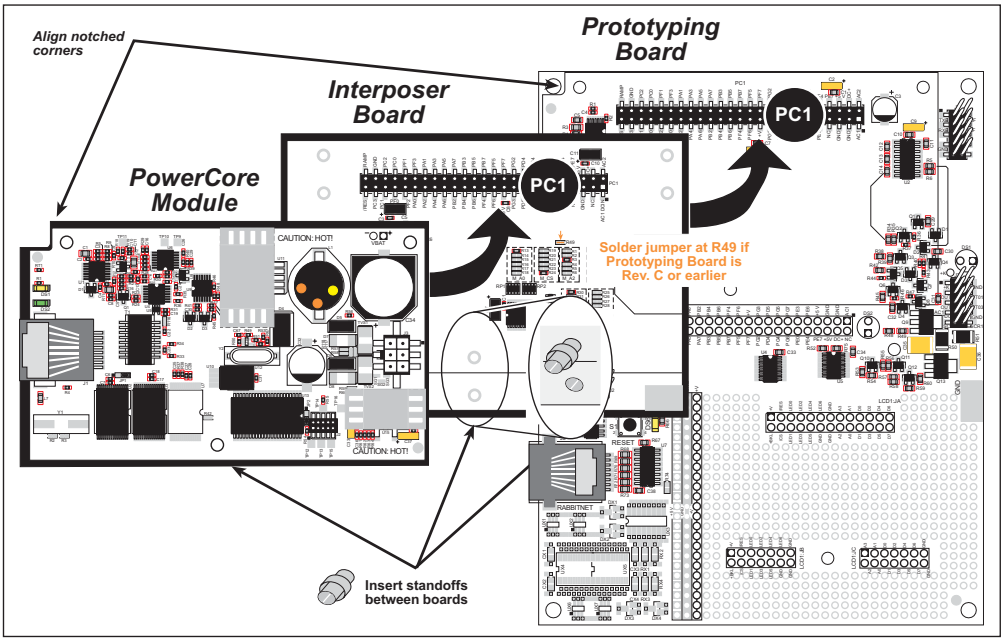


Figure 9. Install the PowerCore Module and Interposer Board on the Prototyping Board

NOTE: It is important that you line up the pins on the PowerCore module and the Interposer Board exactly with the corresponding pins of the sockets. The header pins may become bent or damaged if the pin alignment is offset, and the boards will not work. Permanent electrical damage to the modules may also result if a misaligned module is powered up.

Use additional standoffs to secure the PowerCore module and the Interposer Board to the Prototyping Board in high-vibration environments. Insert the additional standoffs in the two remaining holes near the top on each side of the PowerCore module and the Interposer Board.

2. Connect CompactFlash Wi-Fi Board

Now orient the CompactFlash Wi-Fi Board as shown in Figure 10, and use the 20-pin ribbon cable supplied with the Add-On Kit to connect header J1 of the CompactFlash Wi-Fi Board to header J2 of the Interposer Board. Line up the colored (usually red) edge of the cable towards pin 1 of the two headers. Then insert the Wi-Fi CompactFlash card into the CompactFlash Wi-Fi Board as shown.

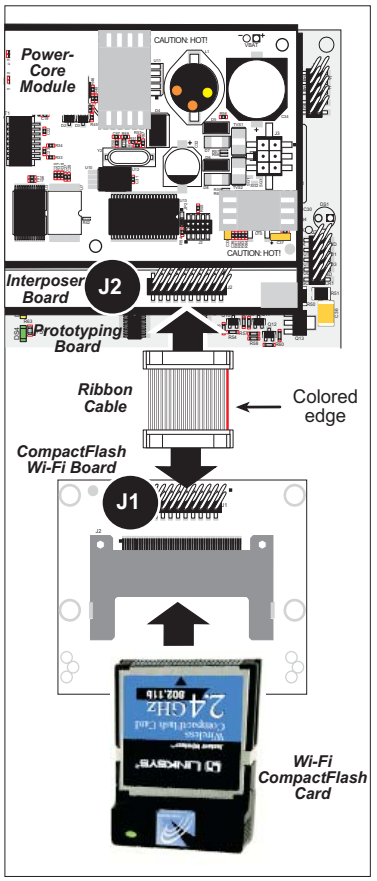


Figure 10. Connect CompactFlash Wi-Fi Board

3. Connect Programming Cable

Connect the 10-pin connector of the programming cable labeled **PROG** to header J2 on your PowerCore module. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the header. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

NOTE: Be sure to use the programming cable supplied with your PowerCore module. Programming cables from other Rabbit Semiconductor kits might not be compatible.

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 540-0070) with the programming cable supplied with the Tool Kit. Note that not all RS-232/USB converters work with Dynamic C.

4. Connect Power Supply

Finally, connect the wall transformer to the J3 locking connector on the PowerCore, taking care to line up the polarized tab on the friction-lock plug with the friction-lock connector as shown in Figure 11 below. Plug in the wall transformer.

NOTE: The **RESET** button is provided on the Prototyping Board to allow a hardware reset without disconnecting power.

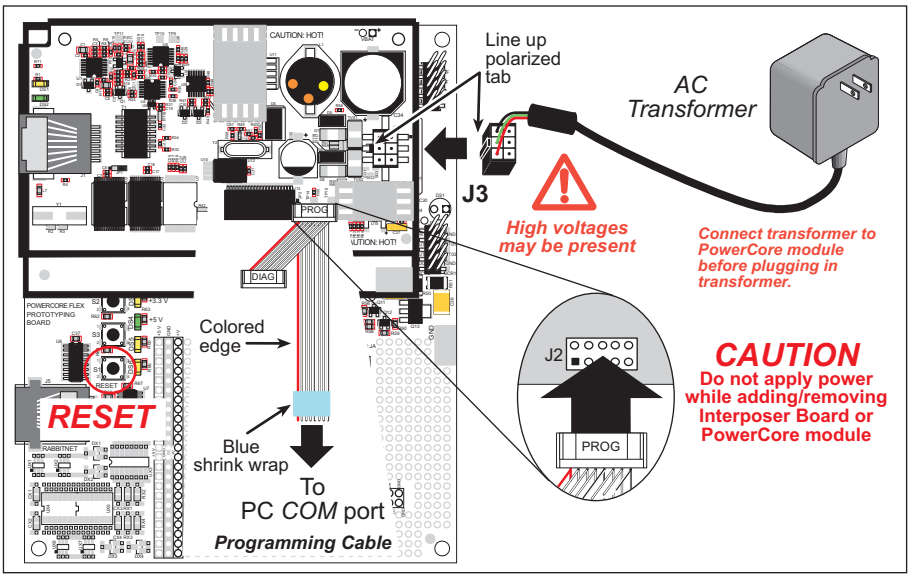


Figure 11. Connect Programming Cable and Power Supply

2.3 Starting Dynamic C

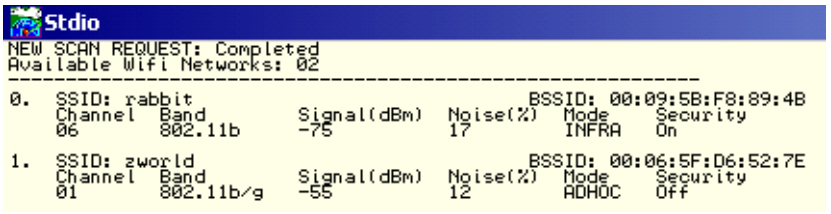
Once the RabbitCore or PowerCore module and the Interposer Board are connected as described in the preceding pages, start Dynamic C by double-clicking on the Dynamic C icon or by double-clicking on **dcrab_XXXX.exe** in the Dynamic C root directory, where **XXXX** are version-specific characters. Dynamic C uses the serial port on your PC that you specified during installation.

If you are using a USB port to connect your computer to the PowerCore module, choose **Options > Project Options** and select "Use USB to Serial Converter."

2.4 Run a Sample Program

Find the **WIFI_SCAN.C** sample program in the Dynamic C **Samples\TCPIP\WiFi** folder, open it with the **File** menu, then compile and run the sample program by pressing **F9**.

The Dynamic C **STDIO** window will display **Scanning....Completed**, and will display a list of access points/ad-hoc hosts in your vicinity as in the example shown below.



The following fields are shown in the Dynamic C **STDIO** window.

- SSID—the name of the service set access point/ad-hoc host.
- Channel—the channel assignment that the access point/ad-hoc host is using (00–14).
- Band—the 802.11 protocol supported by the access point/ad-hoc host.
- Signal—the signal strength of the access point/ad-hoc host in dBm $\left[= 10 \times \log \left(\frac{\text{power}}{1\ \text{mW}} \right) \right]$.
- Noise—the average noise of the channel.
- BSSID—the hardware (MAC) address of the BSS responder.
- Mode—whether the access point/ad-hoc host found is in infrastructure or in ad-hoc mode.
- Security—whether the encryption for the access point/ad-hoc host is on or off.

2.4.1 Troubleshooting

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load a sample program, it is possible that your PC cannot handle the higher program-loading baud rate. Try changing the maximum download rate to a slower baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Select a slower Max download baud rate.

If a program compiles and loads, but then loses target communication before you can begin debugging, it is possible that your PC cannot handle the default debugging baud rate. Try lowering the debugging baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Choose a lower debug baud rate.

If Dynamic C cannot find the target system (error message **"No Rabbit Processor Detected."**):

- Check that the RabbitCore or PowerCore module is powered correctly — the power LEDs on the Prototyping Board should be lit when the module is mounted on the Prototyping Board and the wall transformer is plugged in.
- Check to make sure you are using the **PROG** connector, not the **DIAG** connector, on the programming cable.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and the programming port on the RabbitCore or PowerCore module.
- Ensure that the RabbitCore or PowerCore module is firmly and correctly installed in its socket on the Interposer Board.
- Ensure that the Interposer Board is firmly and correctly installed in its socket on the Prototyping Board.
- Select a different COM port within Dynamic C. From the **Options** menu, select **Project Options**, then select **Communications**. Select another COM port from the list, then click OK. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. If Dynamic C still reports it is unable to locate the target system, repeat the above steps until you locate the active COM port.

2.5 Where Do I Go From Here?

If the sample program ran fine, you are now ready to go on to other sample programs and to develop your own applications. The source code for the sample programs is provided to allow you to modify them for your own use. This manual also provides complete hardware reference information and describes the Wi-Fi software function calls.

For advanced development topics, refer to the ***Dynamic C User's Manual*** and the ***Dynamic C TCP/IP User's Manual***, also in the online documentation set.

2.5.1 Technical Support

NOTE: If you purchased your Wi-Fi Add-On Kit through a distributor or through a Rabbit Semiconductor partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Rabbit Semiconductor Technical Bulletin Board at www.rabbit.com/support/bb/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

3. Wi-Fi Overview

Wi-Fi, a popular name for 802.11b, is one of the wireless schemes available in the 802.11 suite conforming to IEEE standards. 802.11b describes the media access and link layer control for a 2.4 GHz implementation, which can communicate at up to 11 Mbits/s. Other standards describe a faster implementation (54 Mbits/s) in the 2.4 GHz band (802.11g) and a 54 Mbits/s implementation in the 5.6 GHz band (802.11a). The adoption of 802.11 has been fast because it's easy to use and the performance is comparable to wire-based LANs.

Wi-Fi (802.11b) is one of the most common and cost-effective implementations currently available. This is the implementation that is used with Rabbit Semiconductor's Wi-Fi Add-On Kits. A variety of Wi-Fi hardware exists, from wireless access points (WAPs), various Wi-Fi access devices with PCI, PCMCIA, CompactFlash, USB and SDIO interfaces, and Wi-Fi devices such as Web-based cameras and print servers.

802.11b can operate in one of two modes—a managed-access mode (a Basic Service Set or BSS, where one access point is connected to a wireless network), called an infrastructure mode, or an unmanaged mode (a peer-to-peer mode or an Independent Basic Service Set or IBSS), called the ad-hoc mode. The 802.11 standard describes the details of how devices access each other in either of these modes.

Equipment configured for 802.11g may also be used with 802.11b.

3.1 Infrastructure Mode

The infrastructure mode is the most common type of wireless LAN (WLAN) configuration. A Wi-Fi network operating in the infrastructure mode generally consists of one or more wireless access points that are connected to a wired network. These access points allow wireless clients to join these wired networks almost as if they were connected directly. Some of these access points can act as routers or firewalls, particularly for home networks. The infrastructure mode is the easiest way to add wireless devices to an existing network.

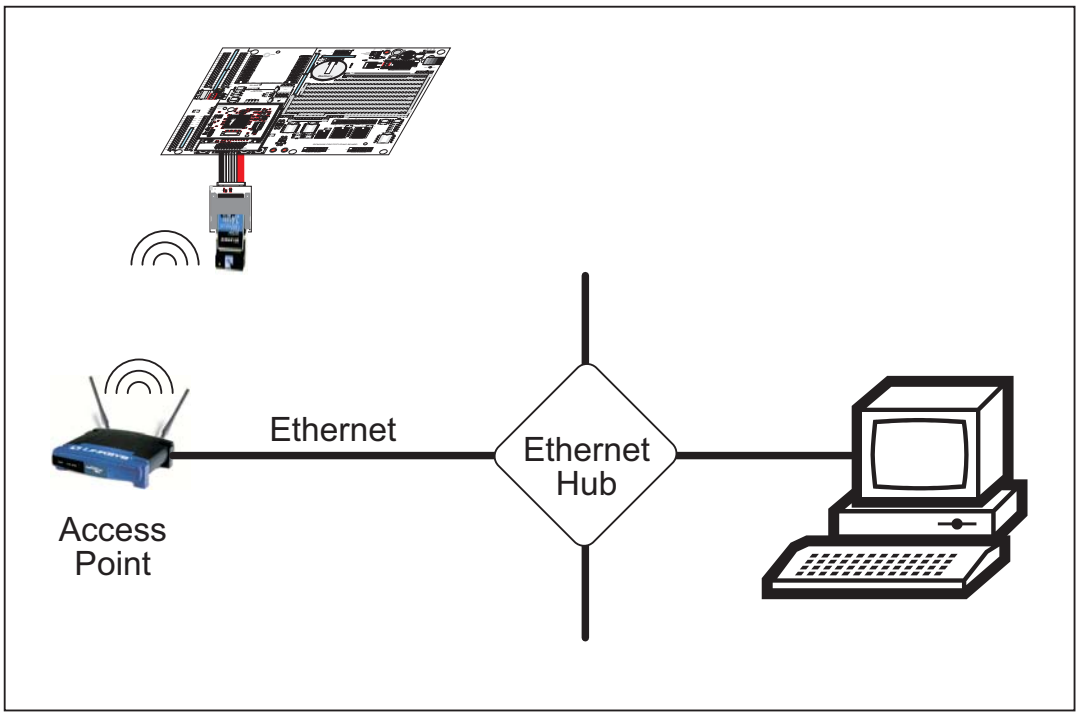


Figure 12. Wi-Fi Network in Infrastructure Mode

To set up a wireless device in an infrastructure mode on a Wi-Fi network, you will need to either assign an IP address to your device or use a DHCP server to assign the IP address for you. You will also need a netmask for your local network, as well as a gateway and name (DNS) server address. Note that these steps are the same as for a wired network.

Some additional configuration is required to enable the device with a wireless capability. First the name of the service set (the SSID or Service Set Identifier) must be assigned to the access point. This name is essentially a name for the WLAN. For example, your SSID might be “rabbitwifi.” This name must be the same for all access points and devices that you want to be on the same wireless network. The SSID may be left blank for the Rabbit device, in which case it will automatically select an access point it detects. Since access points for different WLANs can go up and down, you might connect to an access point you were not expecting. Hence, it is probably best to specify the SSID for all wireless clients.

The other important configuration parameter is wireless encryption (WEP or Wired Equivalent Privacy). In general, you should enable encryption to prevent just any device from joining your WLAN. To enable WEP, you will need to create a common encryption key that must be entered into each access point and wireless client on the WLAN, including the Rabbit-based device. Rabbit-based devices support both 64-bit and 128-bit keys.

Other options, such as authentication and transmit rate can also be set, but the default values for access points and wireless clients (including the Rabbit-based device) should be sufficient.

When a device wants to join an access point, it will typically scan each channel and look for the SSID. Once the access point is found, the device will logically join the access point and announce itself. Once joined, the device can transmit and receive data packets much like in an Ethernet-based network. Being in a joined state is akin to having link status in a 10/100Base-T network.

3.2 Ad-Hoc Mode

The ad-hoc mode allows you to create a less-structured community of wireless devices. No wired or preexisting network is needed. No access point is needed. The wireless devices simply communicate with each other. However, it is difficult (and outside the scope of this manual) to connect an ad-hoc network to other networks (such as the Internet). Ad-hoc networks are most useful for a group of devices that only need to communicate with each other.

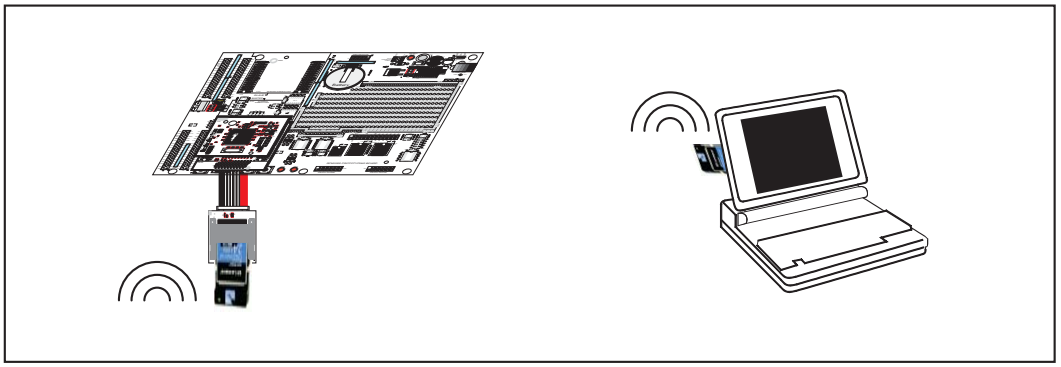


Figure 13. Ad-Hoc Wi-Fi Network

As for the infrastructure mode, a name (the SSID or Service Set Identifier) must be assigned for the ad-hoc network. For a Rabbit-based Wi-Fi network, this name is distinct from the infrastructure SSID, and is referred to as the **`OWNSSID`** in the sample programs and in the software drivers.

A channel must also be specified. In the infrastructure mode, wireless clients detect the channel automatically, but the clients must all be configured to agree on a specific channel when they are being configured in the ad-hoc mode. This channel number may be anything from 0–14, but local regulations in your country may restrict which channels may be assigned. For example, only channels 1–11 are allowed in the United States.

Wireless encryption (WEP) may also be used as in the infrastructure mode. All wireless devices on the network operating in the ad-hoc mode must be programmed with the same encryption key.

3.3 Roaming

A Wi-Fi device may need to roam outside the range of a specific access point in order to perform its functions. Roaming is built into the Wi-Fi CompactFlash cards. As long as certain conditions are met, the Wi-Fi device will be able to roam from one access point to another access point.

1. The access points must all be set to the same SSID.
2. The Rabbit-based device that is roaming must also be set to this SSID.
3. If encryption is used (this is recommended), then all the encryption keys must be set to the same values on all the access points and on the Rabbit-based devices.

The channel may vary from access point to access point—the Rabbit-based device will select the correct channel automatically. The SSID can be left blank when you configure your Rabbit-based device, but then the SSID name it originally connects to will be the only SSID it will accept until your Rabbit-based device is reset. It is probably best to specify the SSID.

The Wi-Fi CompactFlash card will tend to stay connected to the access point that it is currently connected to, even if there is a better signal from another access point. However, once the signal degrades sufficiently, the Wi-Fi CompactFlash card will roam to the better access point.

Note that it is possible for communication to stall for a short period of time as your Rabbit-based device roams. This is because any switches on the network need time to recognize that the Wi-Fi CompactFlash card has moved from one access point to another so that they can update their internal routing tables.

3.4 Additional Information

802.11 Wireless Networking; published by O'Reilly Media, provides further information about 802.11b wireless networks.

4. RUNNING SAMPLE PROGRAMS

To develop and debug Wi-Fi programs for your RabbitCore or PowerCore module (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C.

4.1 Introduction

Dynamic C includes several sample programs to help you understand how to use Wi-Fi with your RabbitCore- or PowerCore-based system. Loading, executing and studying these programs will give you a solid hands-on overview of the module's capabilities, as well as a quick start using Dynamic C as an application development tool.

> **NOTE:** The sample programs assume that you have at least an elementary grasp of the C programming language. If you do not, see the introductory pages of the ***Dynamic C User's Manual*** for a suggested reading list.

In order to run the sample programs discussed in this chapter and elsewhere in this manual,

1. Your RabbitCore or PowerCore module and Interposer Board must be plugged in to the Prototyping Board as described in Chapter 2, "Getting Started."
2. The CompactFlash Wi-Fi Board must have a Wi-Fi CompactFlash card and must be connected to the Interposer Board via the ribbon cable.
3. Dynamic C must be installed and running on your PC.
4. The programming cable must connect the programming header on the RabbitCore or PowerCore module to your PC.
5. Power must be applied to the Prototyping Board.

Refer to Chapter 2, "Getting Started," if you need further information on these steps.

To run a sample program, open it with the **File** menu, and then compile and run the sample program by selecting **Run** in the **Run** menu (or press **F9**). The RabbitCore or PowerCore module must be connected to a PC using the programming cable.

More complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

The ***Getting Started*** instructions included with your Wi-Fi Add-On Kit show how to set up and program the RabbitCore or PowerCore module. Figure 14 shows how your development setup might look once you're ready to proceed.

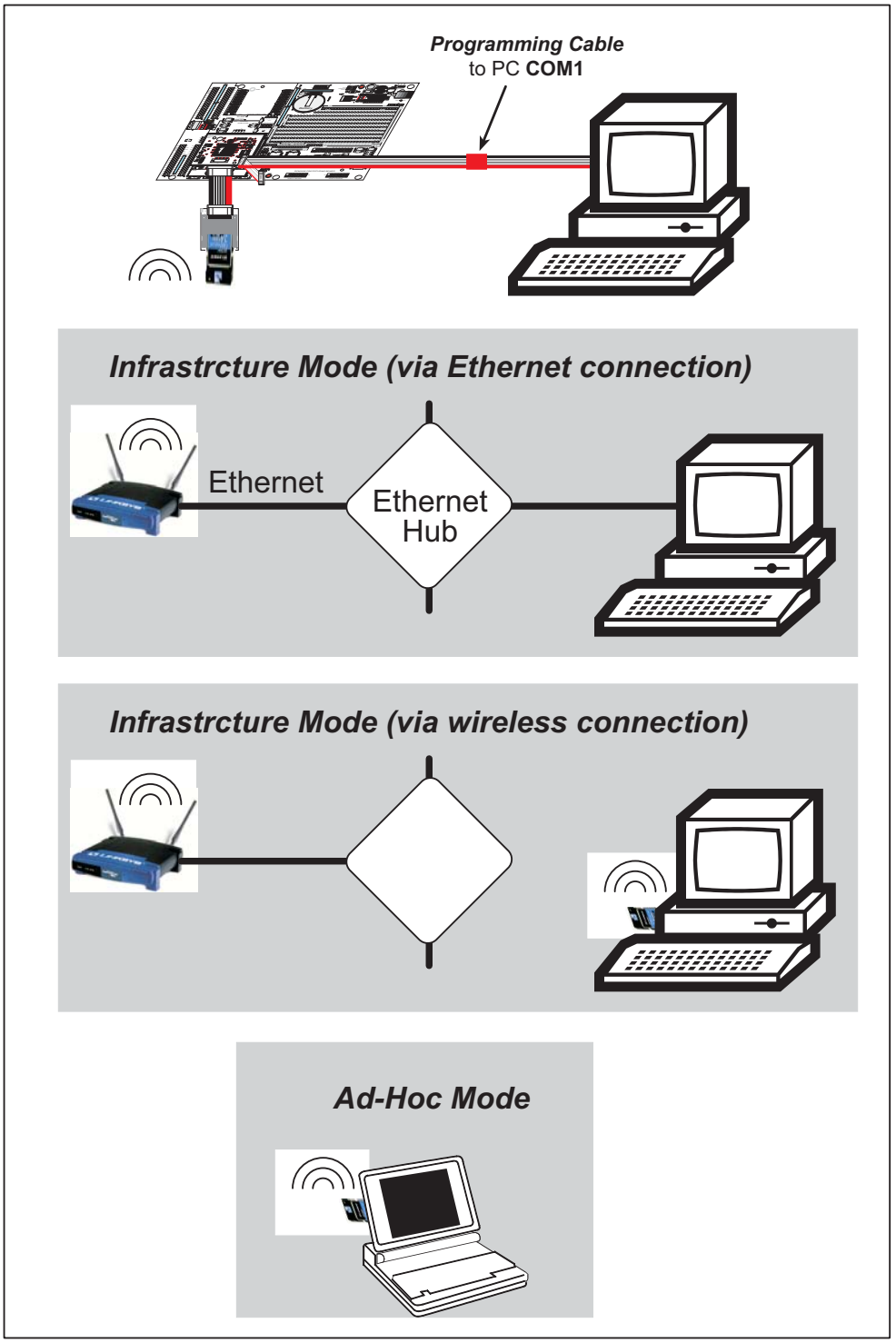


Figure 14. Wi-Fi Setup

4.2 Sample Programs

The following sample programs are available for the Wi-Fi Add-On Kits, and can be found in the Dynamic C **Samples\TCPIP\WIFI** folder.

- **WIFI_SCAN.C**—initializes the Wi-Fi card and scans for other Wi-Fi devices that are operating in either the ad-hoc mode or through access points in the infrastructure mode. No network parameter settings are needed since the Wi-Fi CompactFlash card does not actually join an 802.11b network. This program outputs the results of the scan to the Dynamic C **STDIO** window.
- **WIFI_SCAN_LCD.C**—initializes the Wi-Fi card and scans for other Wi-Fi devices that are operating in either the ad-hoc mode or through access points in the infrastructure mode. No network parameter settings are needed since the Wi-Fi CompactFlash card does not actually join an 802.11b network. This program outputs the results of the scan to the optional LCD/keypad module that plugs into the Prototyping Board.
- **WPINGME.C**—initializes the TCP/IP stack, sets the Wi-Fi card to the infrastructure mode or the ad-hoc mode, and sets the network parameters based on the **TCPCONFIG** macro from the **TCP_CONFIG.LIB** library. The RabbitCore or PowerCore module can then be pinged from another device.
- **DYNAMIC_WIFI.C**—initializes the Wi-Fi CompactFlash card and scans for other Wi-Fi devices that are operating in either the ad-hoc mode or through access points in the infrastructure mode. You will then be able to select which Wi-Fi device/network to connect to.
- **DATALOGGER.C**—logs users who request a Web page via a browser and logs key-presses for switch S2 on the Prototyping Board (switch S1 on the RCM3600/3700 Prototyping Board). The log can be viewed online.
- **WIFISERIAL.C**—allows networked clients to initiate a connection to a TCP/IP socket to communicate with one of the asynchronous serial ports on a Prototyping Board.
- **WIFI_ETHERNET_COMBO.C**—demonstrates how to use the Wi-Fi and wired Ethernet interfaces simultaneously. The sample program shows how to listen for connections on two sockets on two IP addresses specified in the sample program. A Telnet connection to either IP address will echo back each line of typed text.

Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

4.2.1 What Else You Will Need

Besides what is supplied with the Wi-Fi Add-On Kits, you will need a PC with an available COM or USB port to program the RabbitCore or PowerCore module. You will need either an access point for an existing Wi-Fi network that you are allowed to access and have a PC or notebook connected to that network (infrastructure mode), or you will need at least a PDA or PC with a Wi-Fi CompactFlash card to use the ad-hoc mode. If your PC only has a USB port, you will also need an RS-232/USB converter (Part No. 540-0070). You will also need an LCD/keypad module if you wish to run the **WIFI_SCAN_LCD.C** sample program.

4.3 Configuration Information

4.3.1 Network/Wi-Fi Configuration

Any device placed on an Ethernet-based Internet Protocol (IP) network must have its own IP address. IP addresses are 32-bit numbers that uniquely identify a device. Besides the IP address, we also need a netmask, which is a 32-bit number that tells the TCP/IP stack what part of the IP address identifies the local network the device lives on.

The sample programs supplied with the Wi-Fi Add-On Kits configure the RabbitCore or PowerCore modules with a default **TCPCONFIG** macro from the **TCP_CONFIG.LIB** library in the **LIB\TCPIP** directory. This macro allows specific IP address, netmask, gateway, and Wi-Fi parameters to be set at runtime. Change the network settings to configure your RabbitCore or PowerCore with your own Ethernet settings only if that is necessary to run the sample programs; you will likely need to change some of the Wi-Fi settings.

- Network Parameters

  These lines contain the IP address, netmask, nameserver, and gateway parameters.

```
#define _PRIMARY_STATIC_IP "10.10.6.100"
#define _PRIMARY_NETMASK   "255.255.255.0"
#define MY_NAMESERVER      "10.10.6.1"
#define MY_GATEWAY         "10.10.6.1"
```

There are similar macros defined for the various Wi-Fi settings.

- Ad-Hoc Mode or Infrastructure Mode

  The **_WIFI_MODE** macro is used to select the ad-hoc mode (**WIFICONF_ADHOC**) or the infrastructure mode (**WIFICONF_INFRASTRUCT**). The example shows a configuration for the infrastructure mode.

```
#define _WIFI_MODE WIFICONF_INFRASTRUCT
```

- SSID Service Access Point Name

  The **_WIFI_SSID** macro sets the name of the BSS access point you prefer to connect to when operating in the infrastructure mode. The SSID name is a string of 0–32 characters, and restricts the connection to a specific service when specified. The channel is chosen automatically. The example shows the default case where no specific service is specified.

```
#define _WIFI_SSID ""
```

- Wireless Network Settings

  The **_WIFI_OWNCHANNEL** and the **_WIFI_OWNSSID** macros set the name of the IBSS you create, and is usually specified when you plan to use the ad-hoc mode. The SSID name is a string of 1–32 characters, and the channel number is 0–14.

```
#define _WIFI_OWNCHANNEL "10" // Channel "0" through "14"
#define _WIFI_OWNSSID    "RabbitTest" // String specifying your BSS
```

- Encryption

  The **_WIFI_WEP_FLAG** macro is used to enable (**WIFICONF_WEP_ENABLE**) or disable (**WIFICONF_WEP_DISABLE**) key-based encryption. The example shows how to disable encryption.

```
#define _WIFI_WEP_FLAG WIFICONF_WEP_DISABLE
```

- Encryption Keys

  The following four keys can be selected when using WEP encryption. The keys can be modified to match your access point or peer. Shorter keys are supported—you may shorten the key arrays below.

```
#define _WIFI_KEY0 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89

#define _WIFI_KEY1 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89

#define _WIFI_KEY2 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89

#define _WIFI_KEY3 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89
```

  The **_WIFI_USEKEY** specifies which key from the above four keys to use — **"0"** to **"3"** — if you are using WEP encryption. The example shows how to use **KEY0**.

```
#define _WIFI_USEKEY "0"
```

- Authentication Algorithm

  The **_WIFI_AUTH_MODE** macro specifies which authentication algorithm to use when connecting to another wireless service BSS. Use **WIFICONF_AUTH_OPEN_SYS**, **WIFICONF_AUTH_SHARED_KEY**, or **WIFICONF_AUTH_ALL**. These values enable open-system authentication, shared-key authentication, or both, respectively. The most important consideration is to use the same type of authentication as the access point you are planning on using—hence **WIFICONF_AUTH_ALL** is the most flexible value.

```
#define _WIFI_AUTH_MODE WIFICONF_AUTH_ALL
```

Save the library (**File > Save**) once you have made your configuration settings in the Dynamic C **TCP_CONFIG.LIB** library.

The Wi-Fi configurations are contained within **TCPCONFIG 9** (no DHCP) and **TCPCONFIG 10** (with DHCP, used primarily with infrastructure mode). You will need to **#define TCPCONFIG 9** or **#define TCPCONFIG 10** at the beginning of your program.

There are some other "standard" configurations for **TCPCONFIG**. Their values are documented in the **TCP_CONFIG.LIB** library in the **LIB\TCPIP** directory. More information is available in the ***Dynamic C TCP/IP User's Manual***.

4.3.2 PC/Laptop/PDA Configuration

This section shows how to configure your PC or notebook to run the sample programs. Here we're mainly interested in the PC or notebook that will be communicating wirelessly, which is not necessarily the PC that is being used to compile and run the sample program on the RabbitCore or PowerCore module.

This section provides configuration information for the three possible Wi-Fi setups shown in Figure 14. Start by going to the control panel (**Start > Settings > Control Panel**) and click on **Network Connections**. The screen shots shown here are from Windows 2000, and the interface is similar for other versions of Windows.

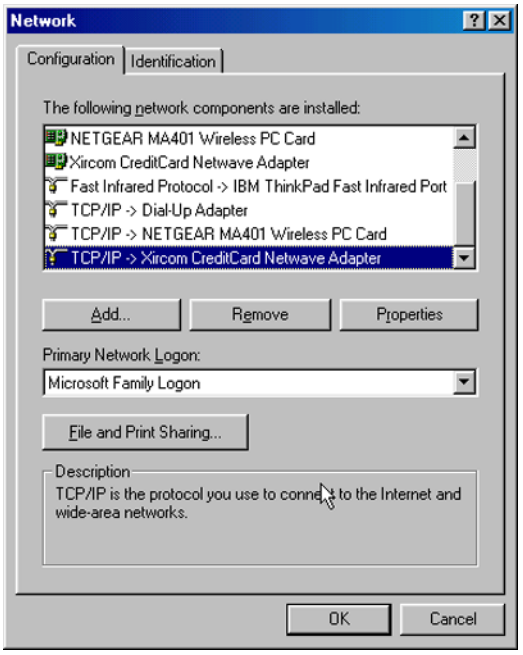
Check with your administrator if you are unable to change the settings as described here since you may need administrator privileges.

When you are using an access point with your setup in the infrastructure mode, you will also have to set the IP address and netmask (e.g., 10.10.6.99 and 255.255.255.0) for the access point. Check the documentation for the access point for information on how to do this.

Infrastructure Mode (via Ethernet connection)

1. Go to the **Local Area Connection** to select the network interface card used you intend to use (e.g., **TCP/IP Xircom Credit Card Network Adapter**) and click on the "Properties" button. Depending on which version of Windows your PC is running, you may have to select the "Local Area Connection" first, and then click on the "Properties" button to bring up the Ethernet interface dialog. Then "configure" your interface card for an "Auto-Negotiation" or "10Base-T Half-Duplex" connection on the "Advanced" tab.

   NOTE: Your network interface card will likely have a different name.



2. Now select the **IP Address** tab, and check **Specify an IP Address**, or select TCP/IP and click on “Properties” to fill in the following fields:

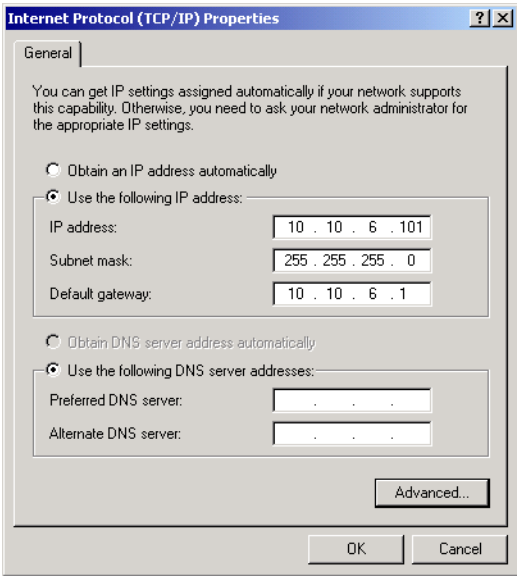
   IP Address : 10.10.6.101

   Netmask : 255.255.255.0

   Default gateway : 10.10.6.1

   TIP: If you are using a PC that is already on a network, you will disconnect the PC from that network to run these sample programs. Write down the existing settings before changing them so that you can restore them easily when you are finished with the sample programs.

   The IP address and netmask need to be set regardless of whether you will be using the ad-hoc mode or the infrastructure mode.



3. Click **<OK>** or **<Close>** to exit the various dialog boxes.

Infrastructure Mode (via wireless connection)

Set the IP address and netmask for your wireless-enabled PC or notebook as described in Step 2 for **Infrastructure Mode (via Ethernet connection)** by clicking on **Network Connections**, then on **Local Area Connection**. Now click on **Wireless Network Connection** to select the wireless network you will be connecting to. Once a sample program is running, you will be able to select the network from a list of available networks. You will have set your wireless network name with the **`_WIFI_SSID`** macro for the infrastructure mode as explained in Section 4.3.1, “Network/Wi-Fi Configuration.”

Ad-Hoc Mode

Set the IP address and netmask for your wireless-enabled PC or notebook as described in Step 2 for **Infrastructure Mode (via Ethernet connection)** by clicking on **Network Connections**, then on **Local Area Connection**. Now click on **Wireless Network Connection** to select the wireless network you will be connecting to. Once a sample program is running, you will be able to select the network from a list of available networks. You will have set your wireless network name with the **`_WIFI_OWNSSID`** and the **`_WIFI_OWNCHANNEL`** macros for the ad-hoc mode as explained in Section 4.3.1, “Network/Wi-Fi Configuration.”

Once the PC or notebook is set up, we're ready to communicate. You can use Telnet or a Web browser such as Internet Explorer, which come with most Windows installations, to use the network interface, and you can use HyperTerminal to view the serial port when these are called for in some of the later sample programs.

Now we're ready to run the sample programs.

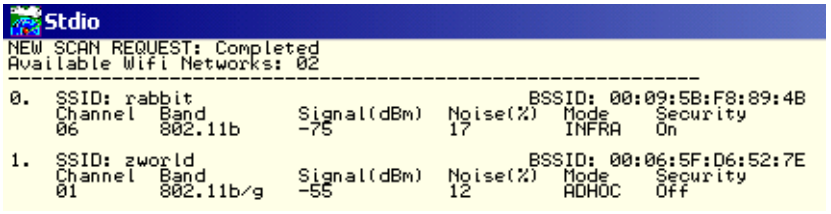
4.4 Sample Programs

WIFI_SCAN.C

WIFI_SCAN.C illustrates the scan function of the Wi-Fi CompactFlash card to look for access points or ad-hoc devices. This sample program does not require additonal configuration of the Wi-Fi CompactFlash card, such as setting the IP address or SSID/channel in order to function, since the Wi-Fi CompactFlash card does not actually join an 802.11b network.

Once the scan request has been issued, the host RabbitCore or PowerCore module will read out the scan results and parse the information. The scan results contain information for each device discovered. This information includes the channel number, the MAC address, the signal strength of the detected device, and the average noise level as seen by the device. It also includes the SSID of the device.

The Dynamic C **STDIO** window will display **Scanning....Completed**, and will display a list of access points/ad-hoc hosts in your vicinity as in the example shown below.



The following fields are shown in the Dynamic C **STDIO** window.

- SSID—the name of the service set access point/ad-hoc host.
- Channel—the channel assignment that the access point/ad-hoc host is using (00–14).
- Band—the 802.11 protocol supported by the access point/ad-hoc host.
- Signal—the signal strength of the access point/ad-hoc host in dBm $\left[= 10 \times \log\left(\frac{\text{power}}{1\ \text{mW}}\right)\right]$.
- Noise—the average noise of the channel.
- BSSID—the hardware (MAC) address of the BSS responder.
- Mode—whether the access point/ad-hoc host found is in infrastructure or in ad-hoc mode.
- Security—whether the encryption for the access point/ad-hoc host is on or off.

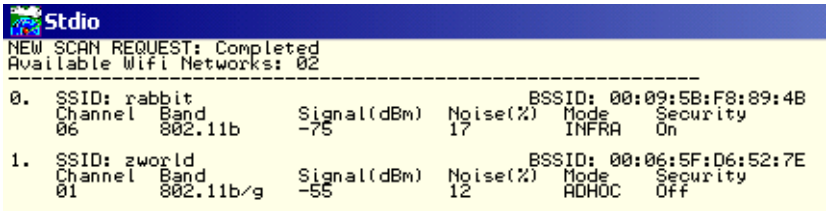
WIFI_SCAN_LCD.C

WIFI_SCAN_LCD.C illustrates the scan function of the Wi-Fi CompactFlash card to look for access points or ad-hoc devices. This sample program does not require additonal configuration of the Wi-Fi CompactFlash card, such as setting the IP number or SSID/channel in order to function, since the Wi-Fi CompactFlash card does not actually join an 802.11b network.

Before running this sample program, you will need to install an LCD/keypad module on the Prototyping Board. Complete instructions are provided in the ***User's Manual*** for the RabbitCore or PowerCore module you are using.

Once you have compiled and run the sample program, and the scan request has been issued, the host RabbitCore or PowerCore module will read out the scan results and parse the information. The scan results contain information for each device discovered. This information includes the channel number, the MAC address, the signal strength of the detected device, and the average noise level as seen by the device. It also includes the SSID of the device.

The Dynamic C **STDIO** window will display **Scanning....Completed**, and will display a list of access points/ad-hoc hosts in your vicinity as in the example shown below.



The same information will scroll down the LCD display on the LCD/keypad module.

WPINGME.C

This sample program initializes the TCP/IP protocol stack and prints out Wi-Fi status information to allow you to determine if your configuration can join an access point or ad-hoc network. The sample program allows pinging from a remote host on the network.

To run this sample program, you will need an access point and PC/notebook (infrastructure mode) or a notebook with 802.11b wireless compatibility working in the ad-hoc mode. By default, encryption is turned off by the **_WIFI_WEP_FLAG** flag in the **TCPCONFIG 9** macro used in the sample program and defined in the Dynamic C **TCP_CONFIG.LIB** library, and the **_WIFI_MODE** macro is set to the infrastructure mode. If you are using the ad-hoc mode, you will need to change **WIFICONF_INFRASTRUCT** to **WIFICONF_ADHOC** in the following line in the Dynamic C **TCP_CONFIG.LIB** library.

```
#define _WIFI_MODE WIFICONF_ADHOC
```

If encryption is turned on, you will have to set the **WIFICONF_WEP_DISABLE** flag to **WIFICONF_WEP_ENABLE** in the following line in the Dynamic C **TCP_CONFIG.LIB** library to enable encryption.

```
#define _WIFI_WEP_FLAG WIFICONF_WEP_ENABLE
```

You may also need to change the encryption keys and select one of the four keys with **#define _WIFI_USEKEY " "**. Contact the administrator of the access point/ad-hoc host or review the documentation for the access point/ad-hoc host to determine the encryption keys.

Once you're ready, just compile and run the sample program. The Dynamic C **STDIO** window will display the Wi-Fi status.

Next, open a command window on a remote host connected to the access point/ad-hoc host, and type the command

```
ping <PRIMARY_STATIC_IP>
```

where **<PRIMARY_STATIC_IP>** is the IP address (10.10.6.100 or whatever IP address you changed it to) set by **_PRIMARY_STATIC_IP** in the **TCPCONFIG 9** macro from the **TCP_CONFIG.LIB** library. One or more reply lines will then appear to display the results of the ping.

DYNAMIC_WIFI.C

This sample program initializes the Wi-Fi CompactFlash card and scans for other Wi-Fi devices that are operating in either the ad-hoc mode or through access points in the infrastructure mode. You will then be able to select which Wi-Fi device/network to connect to.

The **TCPCONFIG 0** macro used in the sample program allows specific IP address, netmask, gateway, and other network parameters to be set at runtime.

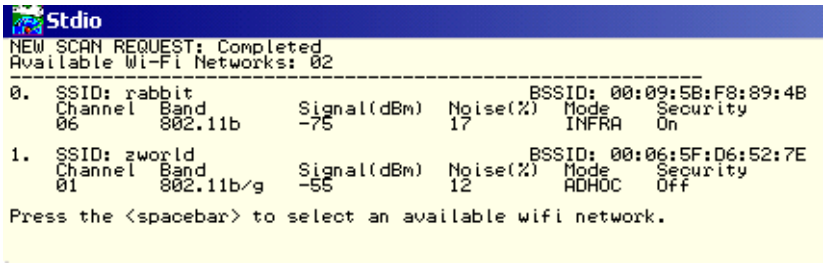
```
#define TCPCONFIG 0 // do not use tcp_config.lib, setting up at runtime

/*
 * The MY_IP_ADDR and MY_GATEWAY_ADDR macros define the IP address and default
 * Gateway address to fallback to, in case DHCP doesn't assign us these values.
 */

#define MY_IP_ADDR "10.10.6.100"
#define MY_GATEWAY_ADDR "10.10.6.1"

#define USE_DHCP   // bring in DHCP
```

When you compile and run this sample program, the Dynamic C **STDIO** window will open to display the available access points and ad-hoc hosts much like the **WIFI_SCAN.C** sample program.



Press the spacebar to get a prompt to select one of the available networks.

```
Select an available network from above 0 thru 1 then press <ENTER>:
```

To connect to rabbit, enter **0**, then press **<Enter>**. The following might then appear in the Dynamic C **STDIO** window.

```
Attaching to "rabbit" in Infrastructure mode (Secured)
This is a WEP network so we will apply our 128 bit key.
Requesting an IP Address. (will use fallbacks if necessary)
DHCP request failed. Using fallback IP address defined in MY_IP_ADDR.
Ping me at: 10.10.6.100
Press <spacebar> to bring down interface and reselect a wifi network.
```

In this case, the request to use DHCP was not successful, and no connection to the wireless network resulted. This particular failure resulted because the encryption key in the sample program was not valid for the rabbit network, and this is a secure network.

Let’s try changing the encryption key in the sample program from

```
const char wifikey[13] = { 0xCC, 0xA1, 0xF2, 0x70, 0x7A, 0xB9, 0xDE, 0xD1,
   0xE6, 0x92, 0xD3, 0xEE, 0x09 };
```

to

```
const char wifikey[13] = { 0xb4, 0x07, 0x83, 0xcb, 0x6d, 0x5a, 0x9f, 0xe3,
   0x8c, 0x38, 0xc0, 0xdf, 0x1c };
```

The Dynamic C **STDIO** window now indicates a successful DHCP request.

.

```
Attaching to "rabbit" in Infrastructure mode (Secured)
This is a WEP network so we will apply our 128 bit key.
Requesting an IP Address. (will use fallbacks if necessary)

DHCP request successful.
Ping me at: 10.10.6.33
Press <spacebar> to bring down interface and reselect a wifi network.
```

DATALOGGER.C

This sample program illustrates the use of embedded Web servers and downloadable data via wireless by implementing a prototype wireless datalogger. A PC or notebook can display or download the log file, reset the log files, and set the current date/time via a Web browser.

To run this sample program, you will need an access point and PC/notebook (infrastructure mode) or a notebook with 802.11b wireless compatibility working in the ad-hoc mode. By default, encryption is turned off by the **_WIFI_WEP_FLAG** flag in the **TCPCONFIG 9** macro used in the sample program and defined in the Dynamic C **TCP_CONFIG.LIB** library, and the **_WIFI_MODE** macro is set to the infrastructure mode. If you are using the ad-hoc mode, you will need to change **WIFICONF_INFRASTRUCT** to **WIFICONF_ADHOC** in the following line in the Dynamic C **TCP_CONFIG.LIB** library.

```
#define _WIFI_MODE WIFICONF_ADHOC
```

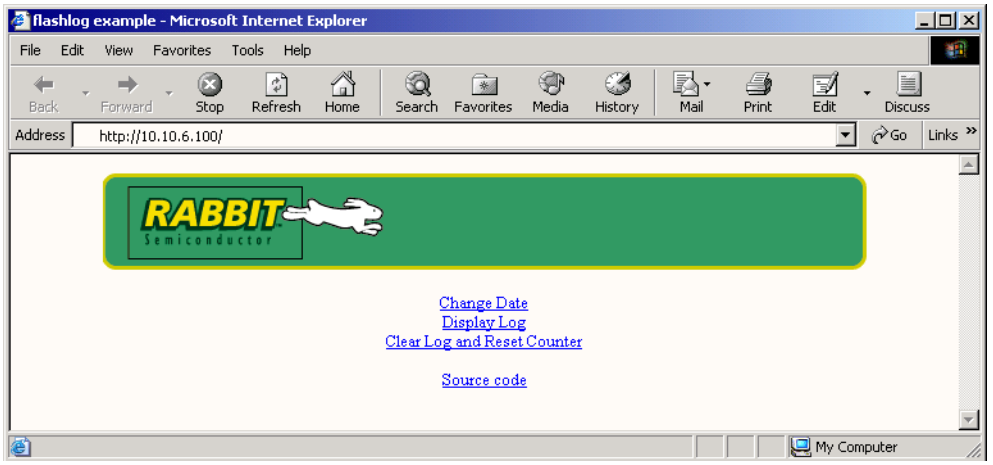
If your access point has encryption turned on for the infrastructure mode, you will have to set the **WIFICONF_WEP_DISABLE** flag to **WIFICONF_WEP_ENABLE** in the following line in the Dynamic C **TCP_CONFIG.LIB** library to enable encryption.

```
#define _WIFI_WEP_FLAG WIFICONF_WEP_ENABLE
```

You may also need to change the encryption keys and select one of the four keys with **#define _WIFI_USEKEY " "**. Contact the administrator of the access point or review the documentation for the access point to determine the encryption keys.

Once the sample program is running either in the infrastructure mode or the ad-hoc mode, it logs the IP address of users who request the Web page via a browser and logs keypresses for switch S2 on the Prototyping Board (switch S1 on the RCM3600/3700 Prototyping Board). The log can be viewed online.

Enter **http://10.10.6.100** (or the network IP address you supplied to the Dynamic C **TCP_CONFIG.LIB** library) and press **Enter** to view the Web page.



The output from this sample program in your Web browser will resemble the following sample output.

Web Log

12/20/2004 11:01:50 - Switch Pressed
12/20/2004 11:01:59 - Switch Pressed
12/20/2004 13:34:14 - 10.10.6.101
12/20/2004 13:36:41 - Switch Pressed
12/20/2004 14:42:06 - Switch Pressed

The Change Date link in the Web browser can be used to reset the date and time for subsequent data logging.

The Clear Log and Reset Counter link in the Web browser is used to clear the log and start it fresh.

WIFISERIAL.C

This sample program is a Wi-Fi to serial converter. It allows networked clients to initiate a connection to a socket and transparently communicate with one of the asynchronous serial ports available on the Prototyping Board.

To run this sample program, you will need an access point and PC/notebook (infrastructure mode) or a notebook with 802.11b wireless compatibility working in the ad-hoc mode. By default, encryption is turned off by the **_WIFI_WEP_FLAG** flag in the **TCPCONFIG 9** macro used in the sample program and defined in the Dynamic C **TCP_CONFIG.LIB** library, and the **_WIFI_MODE** macro is set to the infrastructure mode. If you are using the ad-hoc mode, you will need to change **WIFICONF_INFRASTRUCT** to **WIFICONF_ADHOC** in the following line in the Dynamic C **TCP_CONFIG.LIB** library.

```
#define _WIFI_MODE WIFICONF_ADHOC
```

If your access point has encryption turned on for the infrastructure mode, you will have to set the **WIFICONF_WEP_DISABLE** flag to **WIFICONF_WEP_ENABLE** in the following line in the Dynamic C **TCP_CONFIG.LIB** library to enable encryption.

```
#define _WIFI_WEP_FLAG WIFICONF_WEP_ENABLE
```

You may also need to change the encryption keys and select one of the four keys with **#define _WIFI_USEKEY " "**. Contact the administrator of the access point or review the documentation for the access point to determine the encryption keys.

Once the sample program is running, you will need to connect a serial port on the Prototyping Board to an available COM port on your PC using the serial cable supplied with the Add-On Kits as shown in Figure 15. If your PC only has one COM port (or one USB port), compile and run **WIFISERIAL.C** first with the programming cable attached, then remove the programming cable, reset the board, and connect the serial cable. The location of the serial port varies according to the specific Prototyping Board you are interfacing. Be sure to line up the colored edge of the serial cable with pin 1 of the RS-232 header as shown.

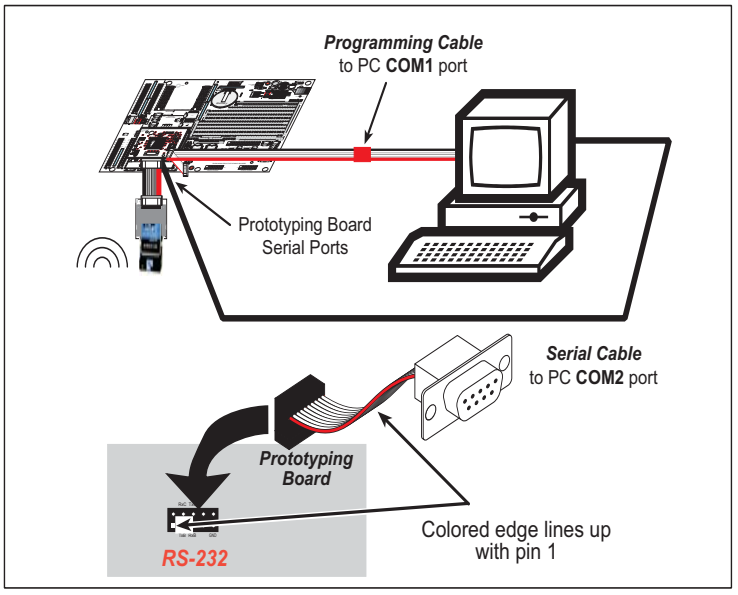


Figure 15. Serial Port Connections

Since the RCM3300 Prototyping Board uses a screw terminal header, you will have to use some hookup wire to connect the serial header to the serial cable as shown in Figure 16.

The locations of the serial headers and their pin 1 locations are shown in Figure 16 for the various Prototyping Boards.

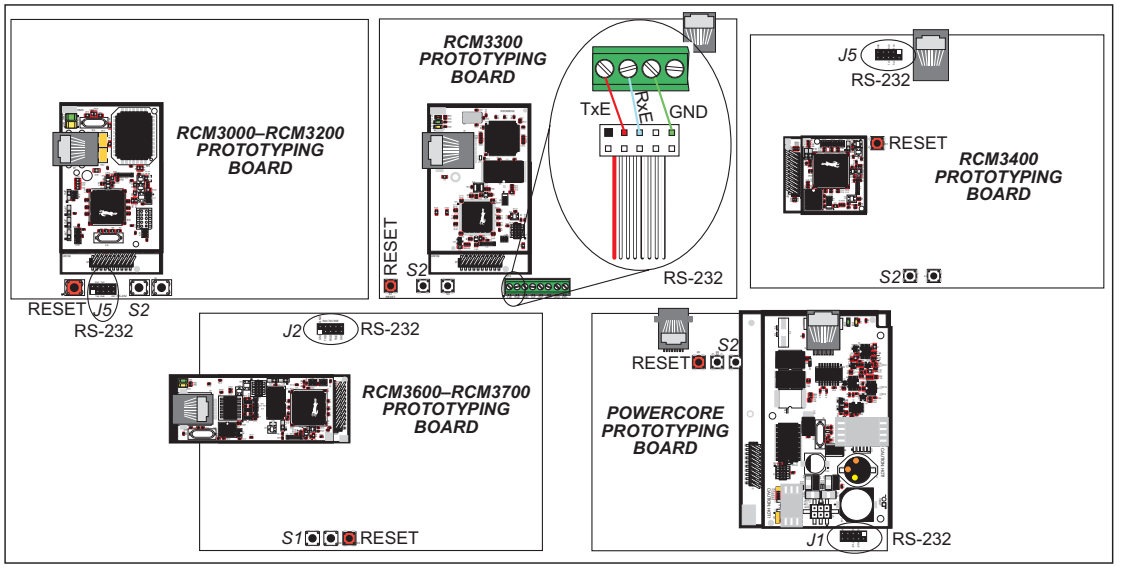


Figure 16. Locations of Serial Headers, Reset Buttons, and Data Logging Pushbutton Switches

Open HyperTerminal on the PC connected to the serial port on the Prototyping Board and set the properties for a COM1 or COM2 connection, depending on which PC serial port is being used.

- 8 data bits
- 1 stop bit
- No parity
- No flow control
- 57600 baud

Next, start a Telnet session (**Start > Run >** `telnet 10.10.6.100`) with the default Telnet port (23) on the remote PC or laptop (use your IP address if you set something other than 10.10.6.100 in **`_PRIMARY_STATIC_IP`** in the **`TCPCONFIG 9`** macro from the **`TCP_CONFIG.LIB`** library). Anything you type in the Telnet window will appear in the HyperTerminal window, and vice versa. Note that there is no character echo, so you will not be able to view the text you are entering in the window where you are typing.

WIFI_ETHERNET_COMBO.C

This sample program shows how to use Wi-Fi and Ethernet interfaces simultaneously. The sample program shows how to listen for connections on two sockets on two IP addresses specified in the sample program. A Telnet connection to either IP address will echo back each character you type in the Telnet window.

To run this sample program, you will need a RabbitCore or PowerCore module with Ethernet. Connect the RJ-45 jack to a PC with a crossover cable or to an Ethernet hub with a straight-through cable. Figure 17 shows how to identify the two cables based on the wires in the transparent RJ-45 connectors.

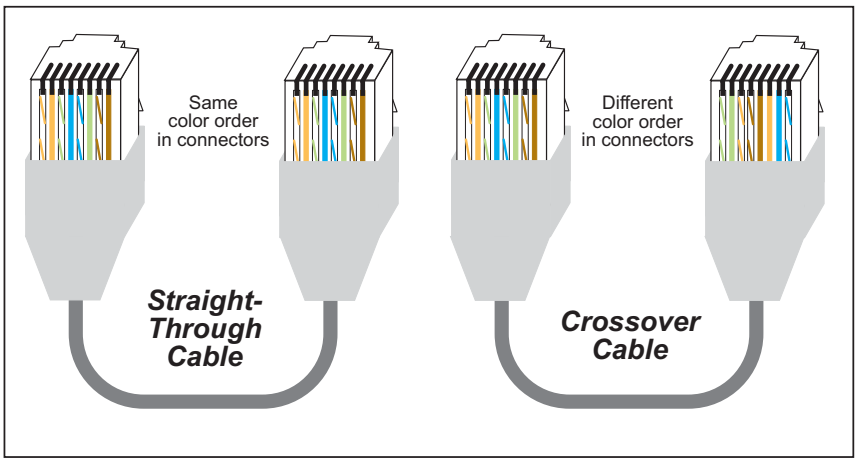


Figure 17. How to Identify Straight-Through and Crossover Ethernet Cables

Ethernet cables and a 10Base-T Ethernet hub are available from Rabbit Semiconductor in a TCP/IP tool kit. More information is available at www.rabbit.com.

You may use the same PC that is running the sample program and is connected to the RabbitCore or PowerCore module via the programming cable. You will have to configure the PC's network interface card and IP address as explained in Section 4.3.2, "PC/Laptop/PDA Configuration." Remember to use a different IP address (for example 10.10.6.102) if you already configured another device with 10.10.6.101.

To run this sample program, you will need an access point and PC/notebook (infrastructure mode) or a notebook with 802.11b wireless compatibility working in the ad-hoc mode. By default, encryption is turned off by the **_WIFI_WEP_FLAG** flag in the **TCPCONFIG 9** macro used in the sample program and defined in the Dynamic C **TCP_CONFIG.LIB** library, and the **_WIFI_MODE** macro is set to the infrastructure mode. If you are using the ad-hoc mode, you will need to change **WIFICONF_INFRASTRUCT** to **WIFICONF_ADHOC** in the following line in the Dynamic C **TCP_CONFIG.LIB** library.

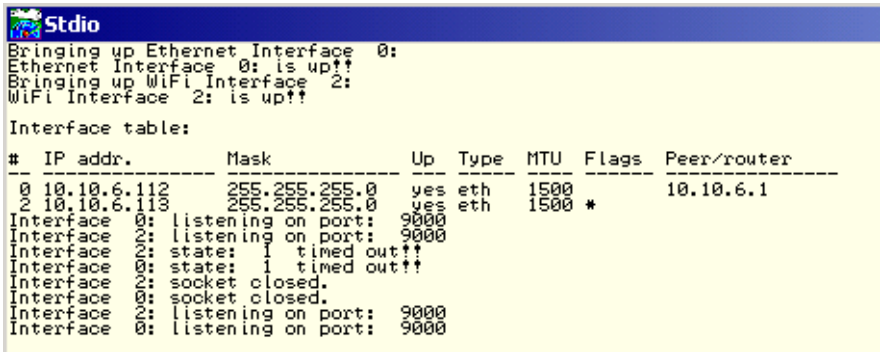
```
#define _WIFI_MODE WIFICONF_ADHOC
```

If your access point has encryption turned on for the infrastructure mode, you will have to set the **WIFICONF_WEP_DISABLE** flag to **WIFICONF_WEP_ENABLE** in the following line in the Dynamic C **TCP_CONFIG.LIB** library to enable encryption.

```
#define _WIFI_WEP_FLAG WIFICONF_WEP_ENABLE
```

You may also need to change the encryption keys and select one of the four keys with **#define _WIFI_USEKEY " "**. Contact the administrator of the access point or review the documentation for the access point to determine the encryption keys.

When you compile and run this sample program, the Dynamic C **STDIO** window will open to display the Ethernet and the Wi-Fi interfaces.



Now you can start a Telnet session to the PC with the Ethernet cable (**Start > Run > telnet 10.10.6.112 9000**) using the default Telnet socket 9000. You will see activity in the Dynamic C **STDIO** window as you type in the Telnet window and your characters are echoed back so you can view them. You can open a command window (**Start > Run > cmd**) and ping the Rabbit to determine that you have a valid Ethernet connection.

```
ping 10.10.6.112
```

Next you can start a Telnet session to the wireless PC or notebook (**Start > Run > telnet 10.10.6.113 9000**) using the default Telnet socket 9000. You will see activity in the Dynamic C **STDIO** window as you type in the Telnet window and your characters are echoed back so you can view them. You can open a command window (**Start > Run > cmd**) and ping the Rabbit to determine that you have a valid Wi-Fi interface.

```
ping 10.10.6.113
```

4.5 Where Do I Go From Here?

Most of the TCP/IP sample programs, including the TCP/IP sample programs specific to your RabbitCore or PowerCore module that are described in the RabbitCore or PowerCore ***User's Manual***, can be run as Wi-Fi sample programs. Just change the **`#define TCPCONFIG`** macro call in the sample program to **`TCPCONFIG 9`**.

```
#define TCPCONFIG 9
```

For advanced development topics, refer to the ***Dynamic C User's Manual*** and the ***Dynamic C TCP/IP User's Manual***, also in the online documentation set.

4.6 Helpful Hints

1. When an access point/ad-hoc host is turned off after a Wi-Fi CompactFlash card has joined it, the Linksys WCF12 card will remain in the "joined" state and will wait for the access point/ad-hoc host to become alive again. This may require cycling of the MAC interface to find a new access point using the following code sequence.

```
...
wifi_ioctl(NULL,WIFI_MAC,"off",0);
wifi_ioctl(NULL,WIFI_MAC,"on",0);
```

2. If you are using a PC or other computer to communicate with the Wi-Fi enabled RabbitCore or PowerCore module, and you change the Wi-Fi CompactFlash card, you may need to flush the ARP cache in the PC or other computer that holds the translation for the hardware (MAC) address to the IP address with the following command at the MS-DOS prompt.

```
arp -d 10.10.6.100
```

   Wi-Fi CompactFlash cards contain their own MAC address, and changing them changes the MAC address for the Wi-Fi enabled embedded application. PCs typically time out the hardware translation cache after a few minutes, or when the interface is restarted. However, it may be necessary to flush the cache manually if you are in a hurry.

3. Do ***not*** hot-swap Wi-Fi CompactFlash cards. The interface and software driver are not designed to allow hot-swapping.

4. The Wi-Fi CompactFlash Board in the Add-On Kits was designed to be used with Wi-Fi CompactFlash cards. It has not been tested for compatibility with other types of CompactFlash cards.

5. SOFTWARE REFERENCE

Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Rabbit Semiconductor controllers and other devices based on the Rabbit microprocessor. Chapter 5 describes the software associated with the Wi-Fi Add-On Kits.

5.1 Dynamic C Functions

Rabbit Semiconductor has implemented a packet driver for PRISM-based 802.11b CompactFlash cards, **CFPRISMINTERP.LIB**, which functions much like an Ethernet driver for the Dynamic C implementation of the TCP/IP protocol stack. In addition to functioning like an Ethernet packet driver, this driver implements an API to access the functions implemented on the 802.11b interface card.

The **CFPRISMINTERP.LIB** driver has been developed around our Wi-Fi Add-On Kits. We have tested this packet driver extensively with the Linksys WCF12 card.

5.1.1 Configuring Dynamic C to Use the CFPRISMINTERP.LIB Driver

The **CFPRISMINTERP.LIB** library is supplied on the supplementary CD-ROM included with the Wi-Fi Add-On Kits. The **CFPRISMINTERP.LIB** and **CFIOINTERP.LIB** libraries will be placed automatically in the Dynamic C **LIB\TCPIP** folder by InstallShield when you install the software and sample programs from the supplementary CD-ROM.

Rabbit Semiconductor has made it easy for you to use the **CFPRISMINTERP.LIB** library by simply adding a **#define** for an already defined **TCPCONFIG** macro from the Dynamic C **TCP_CONFIG.LIB** library at the beginning of your program as in the example below.

```
#define TCPCONFIG 0
```

There are two **TCPCONFIG** macros specifically set up for Wi-Fi applications.

TCPCONFIG 9	Wi-Fi Prism2 chip set, no DHCP
TCPCONFIG 10	Wi-Fi Prism2 chip set, DHCP enabled

These default IP address, netmask, nameserver, and gateway network parameters are set up for the **TCPCONFIG** macros.

```
#define _PRIMARY_STATIC_IP "10.10.6.100"
#define _PRIMARY_NETMASK   "255.255.255.0"
#define MY_NAMESERVER      "10.10.6.1"
#define MY_GATEWAY         "10.10.6.1"
```

These default Wi-Fi parameters are set up for the **TCPCONFIG** macros.

- Infrastructure Mode—**#define _WIFI_MODE WIFICONF_INFRASTRUCT**
- No Access Point SSID—**#define _WIFI_SSID ""**
- Your Own SSID is *RabbitTest*—**#define _WIFI_OWNSSID "RabbitTest"**
- Your Own Channel is 10—**#define _WIFI_OWNCHANNEL "10"**

  Note that there are restrictions on which channels may be used in certain countries. These are provided in Table A-1.
- Encryption is disabled—**#define _WIFI_WEP_FLAG WIFICONF_WEP_DISABLE**
- The following four encryption keys are provided. You will have to modify them according to the encryption keys in effect for the Wi-Fi network you wish to access.

```
#define _WIFI_KEY0 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89
#define _WIFI_KEY1 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89
#define _WIFI_KEY2 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89
#define _WIFI_KEY3 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef, 0x01,
     0x23, 0x45, 0x67, 0x89
```

- Encryption key 0 is selected if encryption is enabled—**#define _WIFI_USEKEY "0"**
- Authentication algorithm enables both open-system authentication and shared-key authentication—**#define _WIFI_AUTH_MODE WIFICONF_AUTH_ALL**

Macros for alternative Wi-Fi configurations are provided later in this chapter, and may be used to change the above default macros or configurations. Alternatively, you may add your own **TCPCONFIG** macro based on these settings as explained in the ***TCP/IP User's Manual***. Save the library (**File > Save**) once you have made your configuration settings in the Dynamic C **TCP_CONFIG.LIB** library.

The **CFIOINTERP.LIB** library supports the **CFPRISMINTERP.LIB** library with I/O functions for the Wi-Fi CompactFlash card, and has no function calls that need to be accessed in a typical customer application.

5.1.2 Function Call

There is one basic function call in the **CFPRISMINTERP.LIB** library.

```
wifi_ioctl(int iface, int cmd, char* data, int len);
```

This function call processes standard Wi-Fi interface configuration and control commands.

PARAMETERS

iface specifies the interface number for the Interposer Board (use **IF_WIFI0**)

This table describes the various requests that can be made for the remaining parameters.

cmd	*data	len	Description
WIFI_SSID	**char***	0–32 characters	Set SSID string
WIFI_MODE	**char***	0	**WIFICONF_INFRASTRUCT** or **WIFICONF_ADHOC**
WIFI_OWNSSID	**char***	1–32 characters	Set SSID string for ad-hoc mode
WIFI_OWNCHAN	**char***	0	"0" through "14"
WIFI_WEP_FLAG	**char***	0	**WIFICONF_WEP_ENABLE** or **WIFICONF_WEP_DISABLE**
WIFI_WEP_USEKEY	**char***	0	"0" through "3"
WIFI_WEP_KEY0	**char[]**	5 or 13	64-bit or 128-bit key
WIFI_WEP_KEY1	**char[]**	5 or 13	64-bit or 128-bit key
WIFI_WEP_KEY2	**char[]**	5 or 13	64-bit or 128-bit key
WIFI_WEP_KEY3	**char[]**	5 or 13	64-bit or 128-bit key
WIFI_WEP_AUTH	**char***	0	**WIFICONF_AUTH_OPEN_SYS**, **WIFICONF_AUTH_SHARED_KEY** or **WIFICONF_AUTH_ALL**
WIFI_STATUS	**struct wifi_status***	**sizeof(struct wifi_status)**	Status returned in ***data**
WIFI_SCANRES	**_wifi_scanResult**	**sizeof(_wifi_scanResult)**	Scan result returned in ***data**
WIFI_MAC	**char***	0	"on" or "off"
WIFI_SCANREQ	**NULL**	0	**NULL**
WIFI_TX_RATE	**char***	0	**WIFICONF_RATE_1MBPS**, **WIFICONF_RATE_2MBPS**, **WIFICONF_RATE_5_5MBPS** or **WIFICONF_RATE_11MBPS**

In the data column:

char* indicates that data argument is a string, and the **len** field is ignored

char[] indicates that the argument is a character array, and **len** indicates the size

struct * indicates that the argument is a pointer to a **struct**, and **len** indicates the size of the **struct**

If you don't want encryption enabled, do not execute any of the WEP commands in the table.

RETURN VALUE

0 if successful

You would use each command macro in its own **wifi_ioctl()** function call. For example, to name the "rabbit" access point and set a transmit rate of 11 Mbits/s, you would have these two lines of code in your program.

```
int wifi_ioctl(IF_WIFI0, WIFI_SSID, "rabbit", 0);
int wifi_ioctl(IF_WIFI0, WIFI_TX_RATE, WIFICONF_RATE_11MBPS, 0);
```

Let's look at the individual **wifi_ioctl()** commands and their macro options.

WIFI_SSID

An SSID (service set identifier) names a specific wireless LAN or network (WLAN). All devices on a single WLAN must share a common SSID. Set this value to your WLAN's SSID, or leave it blank to allow the Rabbit-based device to select a WLAN automatically. Generally, it is best to explicitly set the SSID so that the device does not join a WLAN that you were not expecting it to join.

WIFI_MODE

Set to infrastructure mode (**WIFICONF_INFRASTRUCT**), which is the most common configuration, or ad-hoc mode (**WIFICONF_ADHOC**). Access points are used with the infrastructure mode. No wireless access points are associated with the ad-hoc mode. This allows devices (such as Rabbit-based devices and notebooks) to communicate with each other directly without an access point.

WIFI_OWNSSID

Sets the SSID name of a device operating in the ad-hoc mode, and is not meaningful with the infrastructure mode.

WIFI_OWNCHAN

This parameter specifies the channel the Wi-Fi card uses for your own network when operating in the ad-hoc mode, and is not meaningful with the infrastructure mode.

WIFI_WEP_FLAG

The encryption flag can have one of two values—enabled (**WIFICONF_WEP_ENABLE**) or disabled (**WIFICONF_WEP_DISABLE**). You can use either 64-bit or 128-bit keys for WEP (Wired Equivalent Privacy).

WIFI_WEP_USEKEY

Indicates which key (0–3) is the default transmission key. The setting may be left at the "0" default.

WIFI_WEP_KEY0–3

These are the secret keys that are programmed into each device on a WLAN to use WEP (Wired Equivalent Privacy). Each of these keys must be entered correctly in order for WEP to work. Both 64-bit and 128-bit keys are supported.

WIFI_AUTH

The authentication option is used to configure different types of authentication that the Wi-Fi device supports. There are three types of authentication that are supported—open-system authentication (**WIFICONF_AUTH_OPEN_SYS**), shared-key authentication (**WIFICONF_AUTH_SHARED_KEY**), or both (**WIFICONF_AUTH_ALL**). The most important consideration is to use the same type of authentication as the access point you are planning on using; hence, **WIFICONF_AUTH_ALL** is the most flexible value.

WIFI_STATUS

This command macro returns the status of the connections (to an access point or peer) into a user-supplied structure. The structure is defined in the Wi-Fi libraries as follows.

```
struct _wifi_status {
   int quality;
   int signal;
   int noise;
   int status;
   int rate;
   unsigned char bssid[6]; // associated hw address
   char ssid[32];          // associated ssid
};
```

quality is a value from 0–92 that is calculated from the signal and noise values. A higher value means higher quality. The status member can have one of the following values:

1 = disables

2 = searching for connection

3 = connected to IBSS (ad-hoc mode)

4 = connected to ESS (infrastructure mode)

5 = out of range.

rate indicates the transmit data rate at which the card is currently operating:

1 = 1 Mbits/s

2 = 2 Mbits/s

4 = 5.5 Mbits/s

8 = 11 Mbits/s

bssid indicates the hardware address of the access point to which the Wi-Fi Compact-Flash card is connected.

ssid indicates the SSID name of the connected access point

WIFI_SCANRES

This command macro returns the scan result into a user-supplied structure. The structure is defined in the Wi-Fi libraries as follows.

```
typedef struct {
   unsigned int len;
   unsigned int rid;
   unsigned int rffu;
   unsigned int reason;
   _wifi_ScanEntry table[4];
}
_wifi_ScanResult;
```

len is a value from 0–4 that indicates how many valid BSS (access points and ad-hoc stations) have been returned by the scan request.

rid and **rffu** are used internally.

reason is set to a non-zero value when the scan request has completed.

table contains a list of all scanned access points and ad-hoc stations found within range.

The format of each entry in **table** is defined as a structure in the Wi-Fi libraries as follows.

```
typedef struct {
   unsigned channel;
   unsigned noiseLevel;
   unsigned signalLevel;
   char bssMAC[6];
   unsigned interval;
   unsigned capability;
   char ssid[34];
   unsigned ratesAvail[5];
   unsigned rate;
}
_wifi_ScanEntry;
```

channel is a value from 1-14 identifying the BSS channel from the scan probe response.

noiseLevel is a value from 0x1B to 0x9A indicating the average noise level of the probe response received during the scan.

signalLevel is a value from 0x1B to 0x9A indicating the signal level at which the probe response was received.

bssMAC is the MAC address of the responding BSS access point or ad-hoc station.

interval is the beacon interval from the probe response (in milliseconds).

capability is a bit-mapped integer indicating the capability information attained from the probe response. It can have a combination of the following values ORed together.

0x0001 ESS (“Infrastructure Mode”)
0x0002 IBSS (“Ad-Hoc Mode”)
0x0004 CF-Pollable
0x0008 CF-Poll Request
0x0010 Encryption (WEP) Enabled
0x0020 Short Preamble
0x0040 PBCC
0x0080 Channel Agility
0xFF00 Reserved

Section 7.3.1.4 of the IEEE 802.11b specifications provides additional information about these values.

ssid is the service set identifier attained from the probe response. The format is the length word, provided in the first two bytes, followed by the SSID string, which has a length of 0–32.

ratesAvail specifies all the data rates that the BSS supports. For more information, refer to Section 7.3.2.2 of the IEEE 802.11b specifications.

rate is the data rate of the scan probe response received. It can have the following values.

0x0A = 1 Mbits/s

0x14 = 2 Mbits/s

0x37 = 5.5 Mbits/s

0x6E = 11 Mbits/s

WIFI_MAC

This option can be used to enable (“on”) or disable (“off”) the Wi-Fi CompactFlash card.

WIFI_SCANREQ

Requests a scan of access points to be conducted, or peers if in ad hoc mode. The result of this scan can be read with the **WIFI_SCANRES** command.

WIFI_TX_RATE

This command macro specifies the maximum transmit rate for the Wi-Fi CompactFlash card. This rate is reduced as necessary depending on the quality of the wireless connection. The options are:

1 Mbits/s **(WIFICONF_RATE_1MBPS**)

2 Mbits/s (**WIFICONF_RATE_2MBPS**)

5.5 Mbits/s (**WIFICONF_RATE_5_5MBPS**)

11 Mbits/s (**WIFICONF_RATE_11MBPS**)

6. INSTALLATION AND MOUNTING GUIDELINES

The CompactFlash Wi-Fi Boards supplied with the Wi-Fi Add-On Kits may be panel-mounted. This chapter describes some mounting considerations and includes detailed mounting instructions.

6.1 Mounting Instructions

6.1.1 Mounting Brackets

You may use either the nylon brackets *or* the metal brackets supplied with the Wi-Fi Add-On Kit, but ***not*** both. Figure 18 shows how the two types of brackets are attached to the CompactFlash Wi-Fi Board.

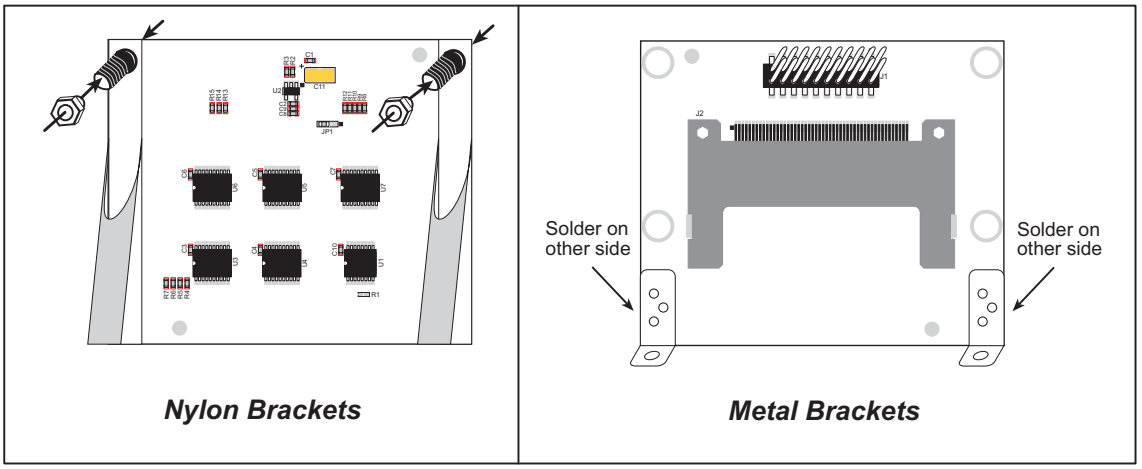


Figure 18. Attach Bracket to Mount Wi-Fi Board

When attaching the nylon brackets, use a 4-40 × 5/16 bolt and locknut with the circular part of the locknut against the nylon bracket as shown in Figure 18. When attaching the right-angle metal brackets, orient their pins with the corresponding holes in the Wi-Fi Board, and solder the pins to the back of the Wi-Fi Board.

6.1.2 Panel Opening and Mounting Holes

Figure 19 shows the opening in the panel for the Wi-Fi CompactFlash card and the two holes to use to mount the CompactFlash Wi-Fi Board with the corresponding bracket option you selected.

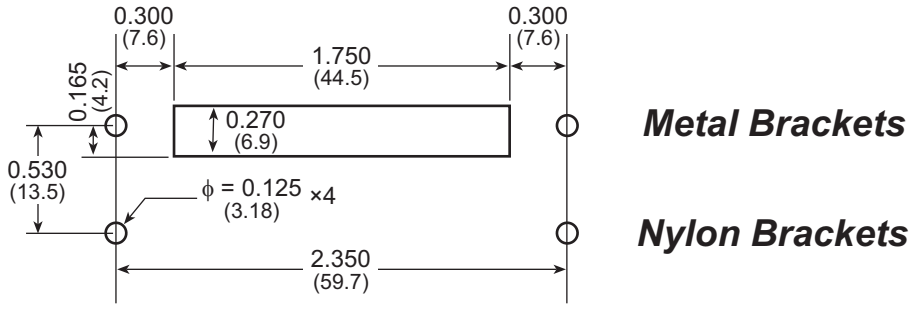


Figure 19. Panel Opening and Mounting Holes for Wi-Fi Board

6.1.3 Attach Wi-Fi Board to Panel

The following step shows you how to mount the CompactFlash Wi-Fi Board on the panel chassis based on which brackets you are using.

1. Nylon Bracket Options—Use either a self-tapping crew or a 4-40 bolt that is at least 3/8" long with a locknut.
2. Metal Bracket—Use a 4-40 bolt that is at least 3/8" long. The metal brackets are already threaded.

These details are illustrated below in Figure 20.

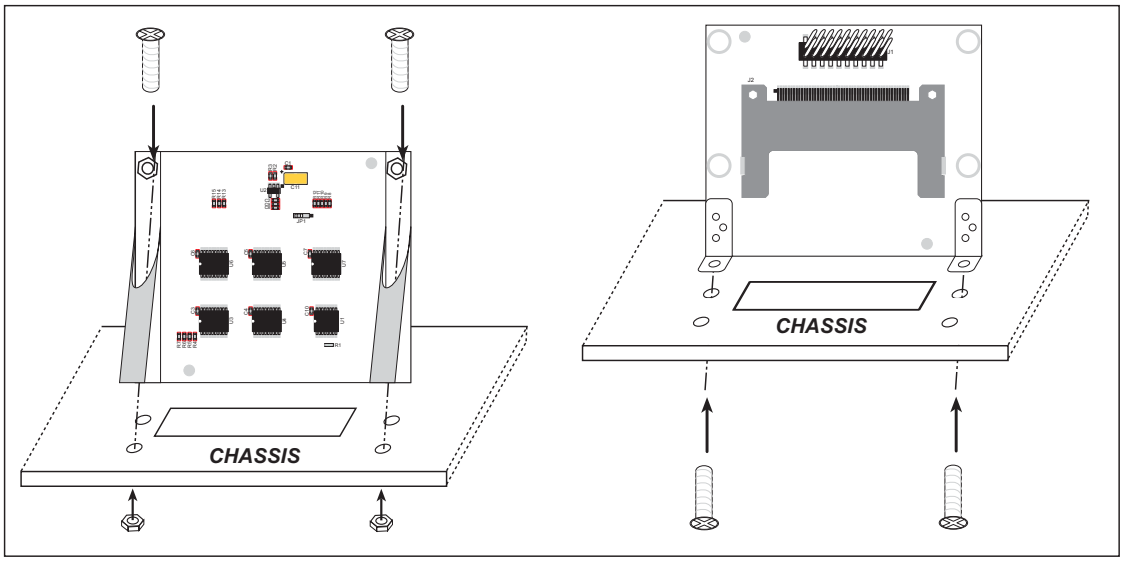


Figure 20. Panel-Mounting Details

Figure 21 shows how the Wi-Fi CompactFlash card is then inserted or removed from the slot in the panel that allows it to be plugged in to the CompactFlash Wi-Fi Board on the other side of the panel.

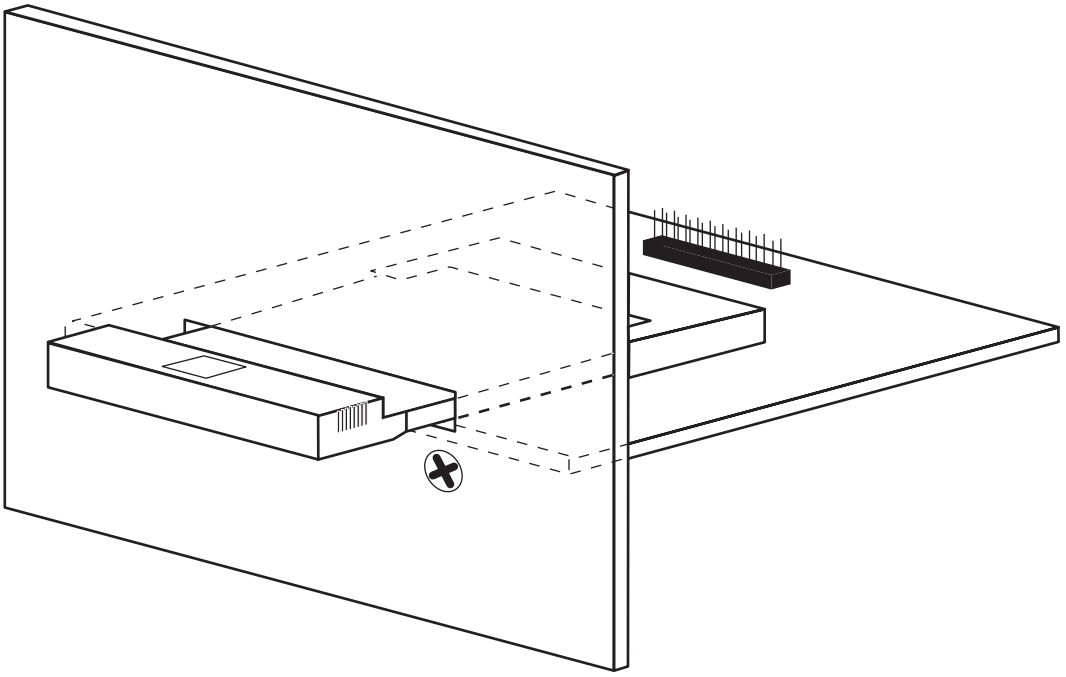


Figure 21. Inserting/Removing Wi-Fi CompactFlash Card

6.2 Grounding

In keeping with good electrical engineering practices, Rabbit Semiconductor recommends that the panel to which the CompactFlash Wi-Fi Board is mounted be connected to an earth ground.

APPENDIX A. SPECIFICATIONS

Appendix A provides the specifications for the Interposer Boards and the CompactFlash Wi-Fi Board included with the various Wi-Fi Add-On Kits.

A.1 Dimensions

A.1.1 CompactFlash Wi-Fi Board

Figure A-1 shows the dimensions for the CompactFlash Wi-Fi Board.

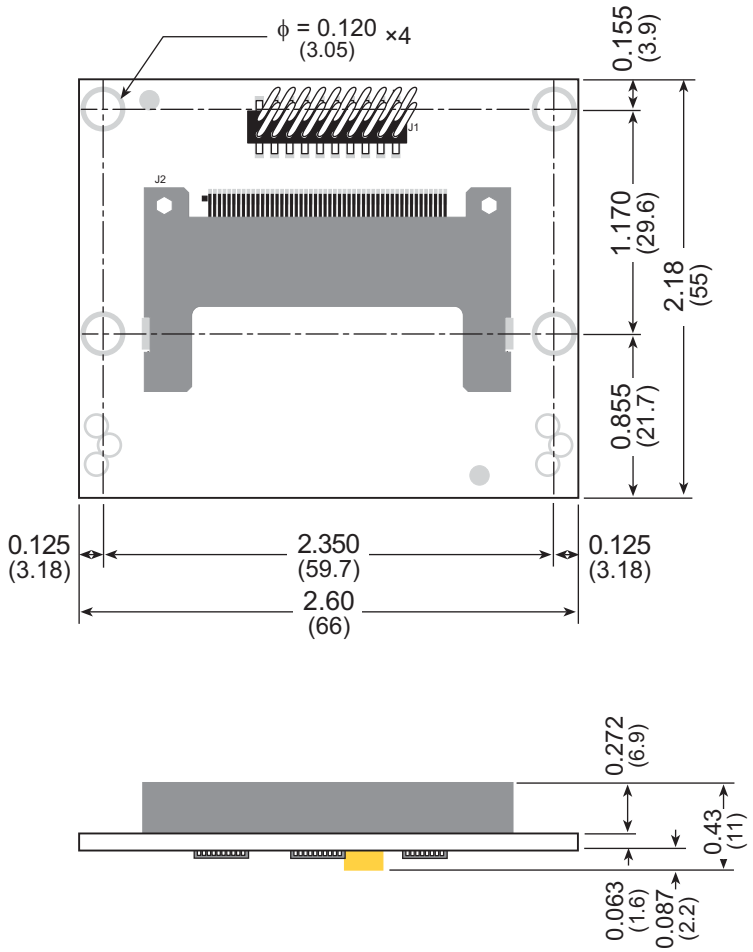


Figure A-1. CompactFlash Wi-Fi Board Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses.

A.1.2 Interposer Boards

A.1.2.1 RCM3000–RCM3300 Interposer Board

Figure A-2 shows the dimensions for the RCM3000–RCM3300 Interposer Board.

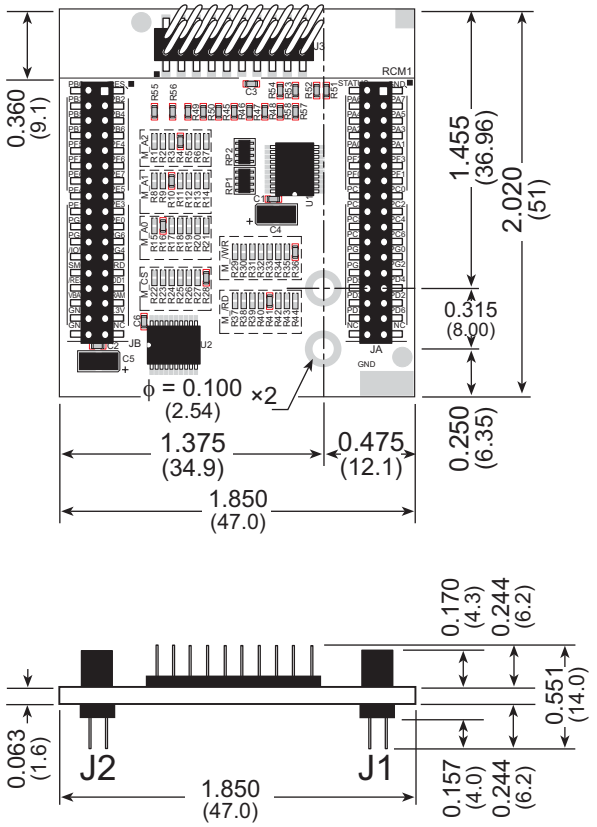


Figure A-2. RCM3000–RCM3300 Interposer Board Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses.

The footprint for the RCM3000–RCM3300 Interposer Board is the same as the footprint for the corresponding RCM3000–RCM3300 RabbitCore module.

A.1.2.2 RCM3400 Interposer Board

Figure A-3 shows the dimensions for the RCM3400 Interposer Board.

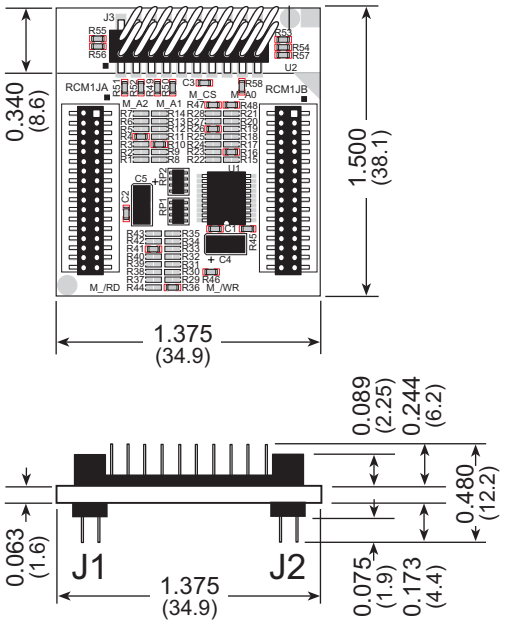


Figure A-3. RCM3400 Interposer Board Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses.

The footprint for the RCM3400 Interposer Board is the same as the footprint for the corresponding RCM3400 RabbitCore module.

A.1.2.3 RCM3600–RCM3700 Interposer Board

Figure A-4 shows the dimensions for the RCM3600–RCM3700 Interposer Board.

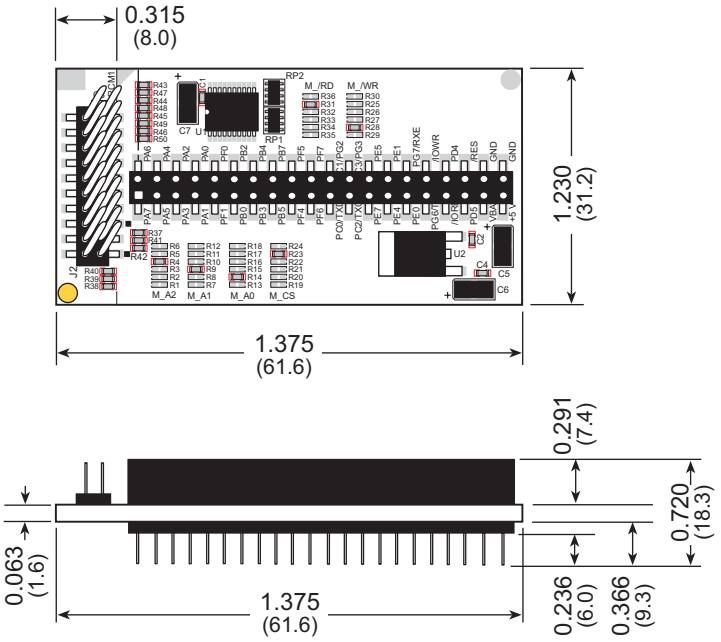


Figure A-4. RCM3600–RCM3700 Interposer Board Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses.

The footprint for the RCM3600–RCM3700 Interposer Board is the same as the footprint for the corresponding RCM3600–RCM3700 RabbitCore module.

A.1.2.4 PowerCore Interposer Board

Figure A-5 shows the dimensions for the PowerCore Interposer Board.

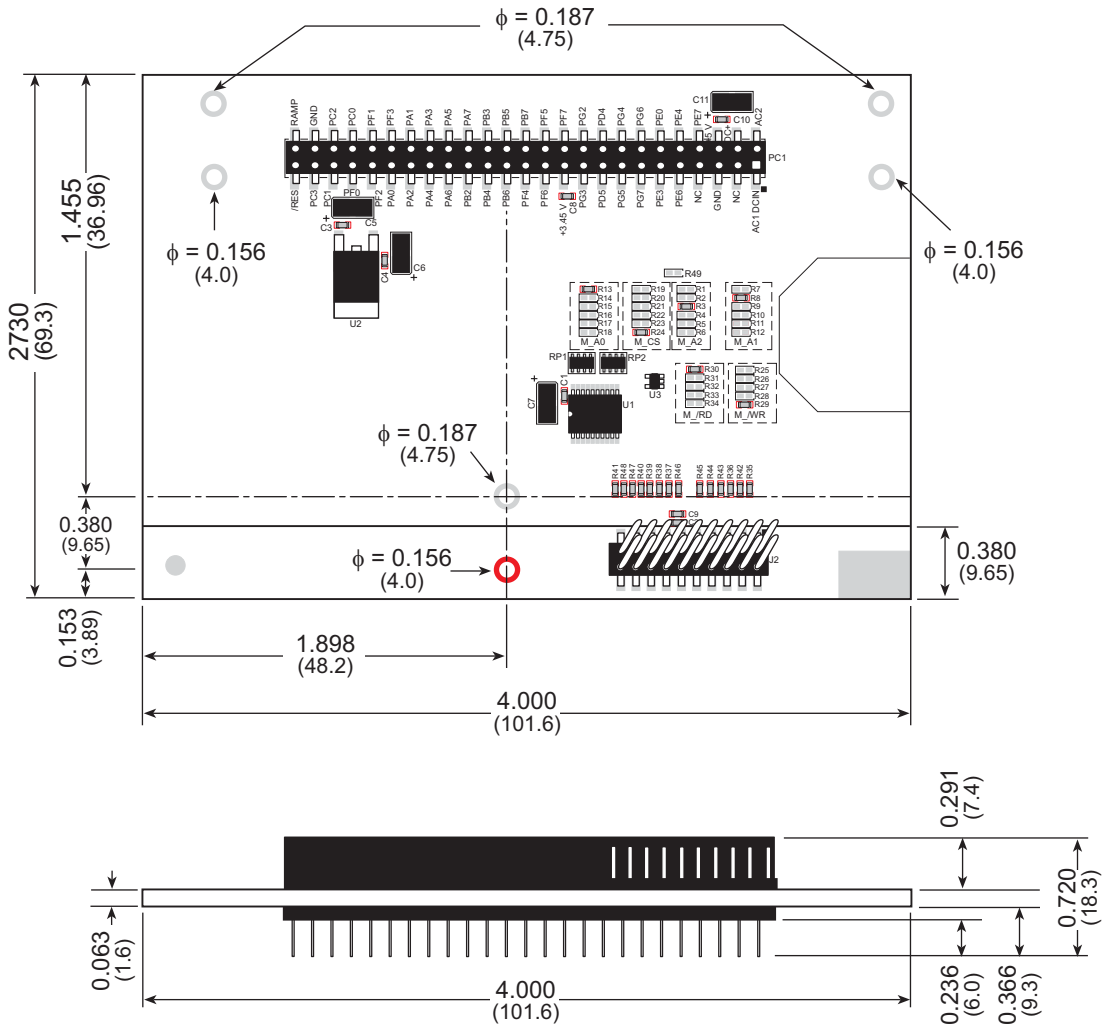


Figure A-5. PowerCore Interposer Board Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses.

NOTE: The diameters of all six holes on PowerCore Interposer Boards sold in 2005 were 0.125" (3.2 mm).

The footprint for the PowerCore Interposer Board is the same as the footprint for the corresponding PowerCore module.

A.2 Electrical, Mechanical, and Wi-Fi Specifications

The following specifications are based on the Linksys WCF12 Wi-Fi CompactFlash card included in the Wi-Fi Add-On Kits.

Table A-1. Wi-Fi Add-On Kits Electrical, Mechanical, and Wi-Fi Specifications

Parameter	Interposer Board			
	RCM3000–RCM3300	RCM3400	RCM3600–RCM3700	PowerCore
Current	285 mA @ 3.3 V	285 mA @ 3.3 V	300 mA @ 5 V	300 mA @ 5 V
Operating Temperature	0°C to +55°C			
Storage Temperature	–20°C to +65°C			
Humidity	10% to 90%, noncondensing			
Wi-Fi CompactFlash Card Certifications	FCC Class B, CE			
Wi-Fi CompactFlash Card Channels	1–11 (Canada, U.S.A) 1–13 (Europe except France, Spain) 10–13 (France) 10–11 (Spain) 14 (Japan)			
Protocol	IEEE 802.11b (Wi-Fi) 11 Mbits/s, 2.4 GHz			
CompactFlash Wi-Fi Board CompactFlash Connector	Type 1 Compact Flash Adapter: • CF based 802.11b I/O • Dual 2 × 10, 2 mm pitch header			

APPENDIX B. CUSTOMIZING INTERPOSER BOARDS

The Interposer Boards used with the Wi-Fi Add-On Kits allow different I/O pins on the Rabbit 3000 microprocessor to be selected for Wi-Fi applications. The actual pins used are set with 0 Ω resistors in hardware, and are assigned with **#define** statements in the **WIFI_INTERP_PINCONFIG.LIB** library. This appendix provides the specific details to enable you to select other available Rabbit 3000 I/O pins for your particular application.

The Rabbit 3000 PA0–PA7 lines are used for external I/O data on all the Interposer Boards. You can reconfigure your Interposer Board to use different Rabbit 3000 I/O pins on Parallel Port B for the address and chip select lines within the pin selections available for each Interposer Board. The Dynamic C **LIB\TCPIP\WIFI_INTERP_PINCONFIG.LIB** library contains the macros for the pin configurations described here. TN106, ***Soldering and Desoldering Surface-Mount Chip Components***, provides instructions on how to safely remove and install the 0 Ω surface-mount resistors used to configure the Interposer Boards.

NOTE: The factory-default settings for the RCM3000–RCM3300 Interposer Board and the RCM3400 Interposer Board use the PB7 line, IA5, for the Wi-Fi M_CS chip select, and they use the /IOWR line as a write strobe. If any external I/O accesses are used in your program with the IA5 address line high on the RCM3000–RCM3300 Interposer Board or the RCM3400 Interposer Board, the Interposer Board may be accessed inadvertently, potentially causing problems for the Wi-Fi operation.

B.1 RCM3000–RCM3300 Interposer Board

Rabbit 3000 I/O	Interposer Board Signal Name	RCM3000–RCM3200 Prototyping Board Signal Name	RCM3300 Prototyping Board Name
PA0	M_D0	ID0	ID0
PA1	M_D1	ID1	ID1
PA2	M_D2	ID2	ID2
PA3	M_D3	ID3	ID3
PA4	M_D4	ID4	ID4
PA5	M_D5	ID5	ID5
PA6	M_D6	ID6	ID6
PA7	M_D7	ID7	ID7
PB0	—	not used	Core SF_CLK
PB2	M_A0	IA0	IA0
PB3	M_A1	IA1	IA1
PB4	M_A2	IA2	IA2
PB5	—	IA3	IA3
PB6	—	not used	not used
PB7	M_CS	not used	not used
PE0	—	not used	IN0
PE1	—	not used	IN1
PE3	—	LCD device select	OUTA CP
PE4	—	IrDA speed select	IN2
PE5	M_/RD	not used	IN3
PE6	—	External I/O strobe	LCD device select
PE7	—	not used	OUTB CP
/IORD	—	/IORD	/IORD
/IOWR	M_/WR	/IOWR	/IOWR

Any of the PB0, PB2, PB3, PB4, PB5, PB6, and PB7 pins on Parallel Port B can be configured for M_A0, M_A1, M_A2, and M_CS via 0 Ω resistors as shown below; the factory defaults are shown in italics.

Rabbit 3000 I/O	M_A0	M_A1	M_A2	M_CS
PB0	R15	R8	R1	R22
PB2	***R16***	R9	R2	R23
PB3	R17	***R10***	R3	R24
PB4	R18	R11	***R4***	R25
PB5	R19	R12	R5	R26
PB6	R20	R13	R6	R27
PB7	R21	R14	R7	***R28***

The assignments from the **WIFI_INTERP_PINCONFIG.LIB** library are shown below for the factory configuration, and would have to be changed according to any changes you make to the pins used on Parallel Port B.

```
#define INTERP0_ADDR0_PORTB_PIN 2 // indicates which PB pin connects to M_A0
#define INTERP0_ADDR1_PORTB_PIN 3 // indicates which PB pin connects to M_A1
#define INTERP0_ADDR2_PORTB_PIN 4 // indicates which PB pin connects to M_A2
#define INTERP0_CS_PORTB_PIN 7 // indicates which PB pin connects to M_CS
```

Any of the PE0, PE1, PE3, PE4, PE5, PE6, and PE7 pins on Parallel Port E can be configured for the M_/RD and M_/WR read and write strobes via 0 Ω resistors as shown below, or /IORD and /IOWR may be used. The factory defaults are shown in italics.

Rabbit 3000 I/O	M_/RD	M_/WR
PE0	R37	R29
PE1	R38	R30
PE3	R39	R31
PE4	R40	R32
PE5	***R41***	R33
PE6	R42	R34
PE7	R43	R35
/IORD	R44	—
/IOWR	—	***R36***

First, the software must identify whether the Parallel Port E read and write strobes are active.

```
#define INTERP0_USE_PORTE_RD_STRB x
#define INTERP0_USE_PORTE_WR_STRB x
```

Use **x** = 0 for strobe on Parallel Port E not active, use **x** = 1 for strobe on Parallel Port E active. If the strobe on Parallel Port E is not active, the corresponding /IORD and /IOWR will be used for the read and write strobes.

The read and write strobe assignments on Parallel Port E are then done in the **WIFI_INTERP_PINCONFIG.LIB** library. The assignments are shown below for the factory configuration, where the read strobe is on pin PE5 of Parallel Port E, and the write strobe has defaulted to /IOWR, which is not on Parallel Port E.

```
#define INTERP0_PORTE_RD_STRB 5 // indicates which PE pin to use
                                // for Read strobe
#define INTERP0_PORTE_WR_STRB 0 // indicates which PE pin to use
                                // for Write strobe
```

The read and write strobe assignments may also be defined in your program, in which case the defaults will be ignored.

The read and write strobe assignments also determine the base address regardless of whether the strobing is being done from Parallel Port E.

Note that if you are using an RCM3300 series RabbitCore with its Prototyping Board, protected digital input IN3 on the RCM3300 Prototyping Board will be unavailable once the Interposer Board is installed because this line will now be used for Wi-Fi read operations.

B.2 RCM3400 Interposer Board

Rabbit 3000 I/O	Interposer Board Signal Name	RCM3400 Prototyping Board Signal Name
PA0	M_D0	ID0
PA1	M_D1	ID1
PA2	M_D2	ID2
PA3	M_D3	ID3
PA4	M_D4	ID4
PA5	M_D5	ID5
PA6	M_D6	ID6
PA7	M_D7	ID7
PB1	—	CLKA
PB2	M_A0	IA0
PB3	M_A1	IA1
PB4	M_A2	IA2
PB5	—	IA3
PB6	—	not used
PB7	M_CS	not used
PE0	—	Optional SMSC interrupt
PE1	—	not used
PE2	—	AEN to SMSC interface
PE4	—	IrDA MIR_SEL
PE5	M_/RD	not used
PE6	—	LCD interface
PE7	—	not used
/IORD	—	/IORD
/IOWR	M_/WR	/IOWR

Any of the PB1, PB2, PB3, PB4, PB5, PB6, and PB7 pins on Parallel Port B can be configured for M_A0, M_A1, M_A2, and M_CS via 0 Ω resistors as shown below; the factory defaults are shown in italics.

Rabbit 3000 I/O	M_A0	M_A1	M_A2	M_CS
PB1	R15	R8	R1	R22
PB2	***R16***	R9	R2	R23
PB3	R17	***R10***	R3	R24
PB4	R18	R11	***R4***	R25
PB5	R19	R12	R5	R26
PB6	R20	R13	R6	R27
PB7	R21	R14	R7	***R28***

The assignments from the **WIFI_INTERP_PINCONFIG.LIB** library are shown below for the factory configuration, and would have to be changed according to any changes you make to the pins used on Parallel Port B.

```
#define INTERP0_ADDR0_PORTB_PIN 2 // indicates which PB pin connects to M_A0
#define INTERP0_ADDR1_PORTB_PIN 3 // indicates which PB pin connects to M_A1
#define INTERP0_ADDR2_PORTB_PIN 4 // indicates which PB pin connects to M_A2
#define INTERP0_CS_PORTB_PIN 7 // indicates which PB pin connects to M_CS
```

Any of the PE0, PE1, PE2, PE4, PE5, PE6, and PE7 pins on Parallel Port E can be configured for the M_/RD and M_/WR read and write strobes via 0 Ω resistors as shown below, or /IORD and /IOWR may be used. The factory defaults are shown in italics.

Rabbit 3000 I/O	M_/RD	M_/WR
PE0	R37	R29
PE1	R38	R30
PE2	R39	R31
PE4	R40	R32
PE5	***R41***	R33
PE6	R42	R34
PE7	R43	R35
/IORD	R44	—
/IOWR	—	***R36***

First, the software must identify whether the Parallel Port E read and write strobes are active.

```
#define INTERP0_USE_PORTE_RD_STRB x
#define INTERP0_USE_PORTE_WR_STRB x
```

Use **x** = 0 for strobe on Parallel Port E not active, use **x** = 1 for strobe on Parallel Port E active. If the strobe on Parallel Port E is not active, the corresponding /IORD and /IOWR will be used for the read and write strobes.

The read and write strobe assignments on Parallel Port E are then done in the **WIFI_INTERP_PINCONFIG.LIB** library. The assignments are shown below for the factory configuration, where the read strobe is on pin PE5 of Parallel Port E, and the write strobe has defaulted to /IOWR, which is not on Parallel Port E.

```
#define INTERP0_PORTE_RD_STRB 5 // indicates which PE pin to use
                                // for Read strobe
#define INTERP0_PORTE_WR_STRB 0 // indicates which PE pin to use
                                // for Write strobe
```

The read and write strobe assignments may also be defined in your program, in which case the defaults will be ignored.

The read and write strobe assignments also determine the base address regardless of whether the strobing is being done from Parallel Port E.

B.3 RCM3600–RCM3700 Interposer Board

Rabbit 3000 I/O	Interposer Board Signal Name	RCM3600–RCM3700 Prototyping Board Signal Name
PA0	M_D0	ID0
PA1	M_D1	ID1
PA2	M_D2	ID2
PA3	M_D3	ID3
PA4	M_D4	ID4
PA5	M_D5	ID5
PA6	M_D6	ID6
PA7	M_D7	ID7
PB0	—	Serial flash SCLK
PB2	M_A0	IA0
PB3	M_A1	IA1
PB4	M_A2	IA2
PB5	M_CS	IA3
PB7	—	Switch S2
PE0	M_/RD	IrDA
PE1	—	IrDA
PE4	M_/WR	IrDA
PE5	—	RS-232
PE7	—	LCD interface
/IORD	—	/IORD
/IOWR	—	/IOWR

Any of the PB0, PB2, PB3, PB4, PB5, and PB7 pins on Parallel Port B can be configured for M_A0, M_A1, M_A2, and M_CS via 0 Ω resistors as shown below; the factory defaults are shown in italics.

Rabbit 3000 I/O	M_A0	M_A1	M_A2	M_CS
PB0	R13	R7	R1	R19
PB2	***R14***	R8	R2	R20
PB3	R15	***R9***	R3	R21
PB4	R16	R10	***R4***	R22
PB5	R17	R11	R5	***R23***
PB7	R18	R12	R6	R24

The assignments from the **WIFI_INTERP_PINCONFIG.LIB** library are shown below for the factory configuration, and would have to be changed according to any changes you make to the pins used on Parallel Port B.

```
#define INTERP0_ADDR0_PORTB_PIN 2 // indicates which PB pin connects to M_A0
#define INTERP0_ADDR1_PORTB_PIN 3 // indicates which PB pin connects to M_A1
#define INTERP0_ADDR2_PORTB_PIN 4 // indicates which PB pin connects to M_A2
#define INTERP0_CS_PORTB_PIN 5 // indicates which PB pin connects to M_CS
```

Any of the PE0, PE1, PE4, PE5, and PE7 pins on Parallel Port E can be configured for the M_/RD and M_/WR read and write strobes via 0 Ω resistors as shown below, or /IORD and /IOWR may be used. The factory defaults are shown in italics.

Rabbit 3000 I/O	M_/RD	M_/WR
PE0	***R31***	R25
PE1	R32	R26
PE4	R33	***R27***
PE5	R34	R28
PE7	R35	R29
/IORD	R36	—
/IOWR	—	R30

First, the software must identify whether the Parallel Port E read and write strobes are active.

```
#define INTERP0_USE_PORTE_RD_STRB x
#define INTERP0_USE_PORTE_WR_STRB x
```

Use **x** = 0 for strobe on Parallel Port E not active, use **x** = 1 for strobe on Parallel Port E active. If the strobe on Parallel Port E is not active, the corresponding /IORD and /IOWR will be used for the read and write strobes.

The read and write strobe assignments on Parallel Port E are then done in the **WIFI_INTERP_PINCONFIG.LIB** library. The assignments are shown below for the factory configuration, where the read strobe is on pin PE0 of Parallel Port E, and the write strobe is on pin PE4 of Parallel Port E.

```
#define INTERP0_PORTE_RD_STRB 0 // indicates which PE pin to use
                                // for Read strobe
#define INTERP0_PORTE_WR_STRB 4 // indicates which PE pin to use
                                // for Write strobe
```

The read and write strobe assignments may also be defined in your program, in which case the defaults will be ignored.

The read and write strobe assignments also determine the base address regardless of whether the strobing is being done from Parallel Port E.

Note that IrDA on the Prototyping Board cannot be used simultaneously with Wi-Fi when the factory configuration of the RCM3600–RCM3700 Interposer Board is used.

B.4 PowerCore Interposer Board

Rabbit 3000 I/O	Interposer Board Signal Name	PowerCore Prototyping Board Signal Name
PA0	M_D0	ID0
PA1	M_D1	ID1
PA2	M_D2	ID2
PA3	M_D3	ID3
PA4	M_D4	ID4
PA5	M_D5	ID5
PA6	M_D6	ID6
PA7	M_D7	ID7
PB2	M_A0	IA0
PB3	M_A1	IA1
PB4	M_A2	IA2
PB5	—	IA3
PB6	—	not used
PB7	M_CS	not used
PE0	M_/RD	Sinking Output
PE3	—	Sourcing Output
PE4	—	RabbitNet RS-422
PE6	—	LCD_CS
PE7	M_/WR	LCD_DIR

Any of the PB2, PB3, PB4, PB5, PB6, and PB7 pins on Parallel Port B can be configured for M_A0, M_A1, M_A2, and M_CS via 0 Ω resistors as shown below; the factory defaults are shown in italics.

Rabbit 3000 I/O	M_A0	M_A1	M_A2	M_CS
PB2	***R13***	R7	R1	R19
PB3	R14	***R8***	R2	R20
PB4	R15	R9	***R3***	R21
PB5	R16	R10	R4	R22
PB6	R17	R11	R5	R23
PB7	R18	R12	R6	***R24***

The assignments from the **WIFI_INTERP_PINCONFIG.LIB** library are shown below for the factory configuration, and would have to be changed according to any changes you make to the pins used on Parallel Port B.

```
#define INTERP0_ADDR0_PORTB_PIN 2 // indicates which PB pin connects to M_A0
#define INTERP0_ADDR1_PORTB_PIN 3 // indicates which PB pin connects to M_A1
#define INTERP0_ADDR2_PORTB_PIN 4 // indicates which PB pin connects to M_A2
#define INTERP0_CS_PORTB_PIN 7 // indicates which PB pin connects to M_CS
```

Any of the PE0, PE3, PE4, PE6, and PE7 pins on Parallel Port E can be configured for the M_/RD and M_/WR read and write strobes via 0 Ω resistors as shown below. The factory defaults are shown in italics.

Rabbit 3000 I/O	M_/RD	M_/WR
PE0	***R30***	R25
PE3	R31	R26
PE4	R32	R27
PE6	R33	R28
PE7	R34	***R29***

First, the software must identify whether the Parallel Port E read and write strobes are active.

```
#define INTERP0_USE_PORTE_RD_STRB x
#define INTERP0_USE_PORTE_WR_STRB x
```

Use **x** = 0 for strobe on Parallel Port E not active, use **x** = 1 for strobe on Parallel Port E active. By default, both the read and write strobes on Parallel Port E should be active.

The read and write strobe assignments on Parallel Port E are then done in the **WIFI_INTERP_PINCONFIG.LIB** library. The assignments are shown below for the factory configuration, where the read strobe is on pin PE0 of Parallel Port E, and the write strobe is on pin PE7 of Parallel Port E.

```
#define INTERP0_PORTE_RD_STRB 0 // indicates which PE pin to use
                                // for Read strobe
#define INTERP0_PORTE_WR_STRB 7 // indicates which PE pin to use
                                // for Write strobe
```

The read and write strobe assignments may also be defined in your program, in which case the defaults will be ignored.

The read and write strobe assignments also determine the base address regardless of whether the strobing is being done from Parallel Port E.

Note that the OUT01 sinking output on the PowerCore Prototyping Board cannot be used simultaneously with Wi-Fi when the factory configuration of the PowerCore Interposer Board is used.

APPENDIX C. COMPACTFLASH CARDS

The following CompactFlash cards were evaluated by Rabbit Semiconductor for the Wi-Fi Add-On Kits, and were found to provide satisfactory results.

Manufacturer and Model	Hardware	Onboard Firmware	Comments
Belkin F5D6060	—	—	Type 1.5
D-Link DCF-660W	v. A1	1.3.6	
Linksys WCF11	—	—	Type 1.5
Linksys WCF12	—	—	
Netgear MA701	v. 1.0	1.4.9	
Pharos PEW-CF11	—	—	
SMC SMC2642W	—	1.3.6	
TRENDnet TEW-222CF	—	1.4.2	

INDEX

T

W

SCHEMATICS

090-0201 CompactFlash Wi-Fi Board Schematic

www.rabbit.com/documentation/schemat/090-0201.pdf

090-0200 RCM3000–RCM3300 Interposer Board Schematic

www.rabbit.com/documentation/schemat/090-0200.pdf

090-0205 RCM3400 Interposer Board Schematic

www.rabbit.com/documentation/schemat/090-0205.pdf

090-0202 RCM3600–RCM3700 Interposer Board Schematic

www.rabbit.com/documentation/schemat/090-0202.pdf

090-0204 PowerCore Interposer Board Schematic

www.rabbit.com/documentation/schemat/090-0204.pdf

You may use the URL information provided above to access the latest schematics directly.