Application Note

AN404

802.11b (Wi-Fi®) Application Kit

Introduction

The 802.11b (Wi-Fi) Application Kit is targeted at experienced embedded systems users to illustrate how to expand the connectivity of Z-World's single-board computers and RabbitCore modules to wireless networks.

What Else You Will Need

Besides what is supplied with the Application Kit, you will need a PC with an available COM or USB port to program the RCM3100 in the Application Kit, and you will also need at least a PDA or laptop that is compatible with the 802.11b wireless standard. If your PC only has a USB port, you will also need an RS-232/USB converter (Z-World Part No. 540-0070).

The ***802.11b (Wi-Fi) Application Kit Getting Started*** instructions included with the Application Kit show how to set up and program the RCM3100. Figure 1 shows how your development setup might look once you're ready to proceed.

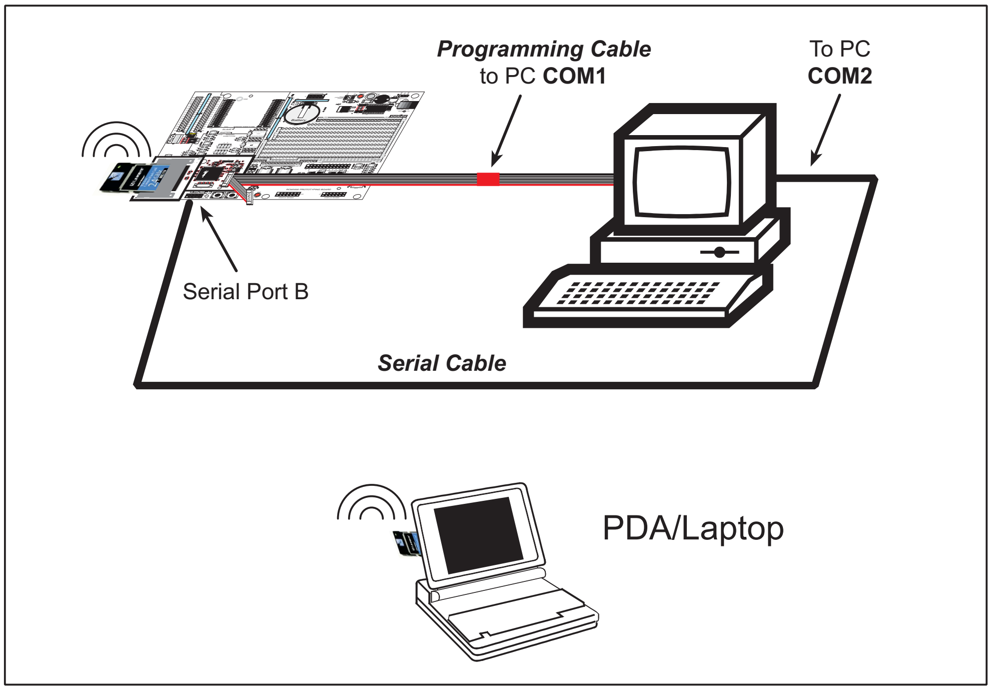


Figure 1. 802.11b Application Kit Wireless Setup

Overview of 802.11b (Wi-Fi)

Wi-Fi[*], a popular name for 802.11b, is one of the wireless schemes available in the 802.11 suite conforming to standards defined by IEEE. 802.11b describes the media access and link layer control for a 2.4 GHz implementation, which can communicate at a top bit-rate of 11 megasamples/s. Other standards describe a faster implementation (54 megasamples/s) in the 2.4 GHz (802.11g) and a 54 megasamples/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. Things look pretty much like a wireless LAN.

Wi-Fi (802.11b) is the most common and cost-effective implementation currently available. This is the implementation that is used with Z-World’s Wi-Fi Application Kit. A variety of Wi-Fi hardware exists, from wireless access points (WAPs), various Wi-Fi access devices with PCI, PCMCIA, CompactFlash, USB and SD/MMC interfaces, and Wi-Fi devices such as Web-based cameras and print servers.

802.11b can operate in one of two modes—in a managed-access mode (BSS), called an infrastructure mode, or an unmanaged mode (IBSS), called the ad-hoc mode. The 802.11 standard describes the details of how devices access each other in either of these modes.

Infrastructure Mode

The infrastructure mode requires an access point to manage devices that want to communicate with each other. An access point is identified with a channel and SSID that it uses to communicate. Typically, an access point also acts as a gateway to a wired network, either an Ethernet or WAN (DSL/cable modem). Most access points can also act as a DHCP server, and provide IP, DNS, and gateway functions.

When a device wants to join an access point, it will typically scan each channel and look for a desired SSID for the access point. A special SSID “default” or “any” means match the SSID of any access point.

Once the access point is discovered, the device will logically join the access point and announce itself. Once joined, the device can transmit and receive data packets much like an Ethernet-based MAC. Being in a joined state is akin to having link status in a 10/100Base-T network.

802.11b interface cards implement all of the 802.11b low-level configurations in firmware. In fact, the 802.11b default configuration is often sufficient for a device to join an access point automatically, which it can do once enabled. Commands issued to the chip set in the interface allow a host program to override the default configurations and execute functions implemented on the interface cards, for example, scanning for hosts and access points.

Ad-Hoc Mode

In the ad-hoc mode, each device can set a channel number and a service set identifier (SSID) to communicate with. If devices are operating on the same channel and SSID, they can talk with each other, much like they would on a wired LAN such as an Ethernet. This works fine for a few devices that are statically configured to talk to each other, and no access point is needed.

Additional Information

802.11 Wireless Networking; published by O'Reilly Media, provides further information about 802.11b wireless networks. The latest version of this application note is available on Z-World’s Web site.

[*] Wi-Fi® and the Wi-Fi logo are registered trademarks of the Wi-Fi Alliance.

Dynamic C Functions

Z-World has implemented a packet driver for PRISM-based 802.11b CompactFlash cards, **`CFPRISM.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 **`CFPRISM.LIB`** driver has been developed around Z-World's 802.11b (Wi-Fi) Application Kit, which includes a CompactFlash Adapter Board, a CompactFlash-based 802.11b wireless network card, a Rabbit-Core RCM3100 module, and a Prototyping Board, a programming cable, and a serial cable. We have tested this packet driver extensively with the Linksys WCF12 card.

Configuring Dynamic C to Use the `CFPRISM.LIB` Driver

The **`CFPRISM.LIB`** driver is supplied on the supplementary CD-ROM included with the 802.11b (Wi-Fi) Application Kit. Copy **`CFPRISM.LIB`** and **`CFIO.LIB`** into the **`Lib\tcpip`** folder of your Dynamic C installation if you have not yet done so, and modify the **`LIB.DIR`** file in the **`Lib`** folder by adding the lines below.

```
LIB\TCPIP\CFPRISM.LIB
LIB\TCPIP\CFIO.LIB
```

This information is provided in the **`WiFiReadMe.txt`** file on the supplementary CD-ROM.

Next, you will need to specify this driver to the TCP/IP protocol stack by adding **`#define PKTDRV`** to your source code as in the example below.

```
...
#define PKTDRV cfprism.lib
#use dcrtcp.lib
...
```

Alternately, you may add this **`#define`** line to the compiler define configuration, which is saved in your project file.

The **`#define PKTDRV`** overrides any packet drivers that may be defined automatically for RabbitCore modules that already have an Ethernet interface, and replaces it with the **`CFPRISM.LIB`** packet driver. If you are using a RabbitCore module such as the RCM3100 without an Ethernet interface, this **`#define PKTDRV`** will select the **`CFPRISM.LIB`** packet driver.

Note that the sample programs on the supplementary CD-ROM do not require the **`#define PKTDRV`** declaration.

You may use all the standard TCP/IP definitions normally used with Ethernet devices as defined in the ***TCP/IP User's Manual***. [Note that power management and multicasting have not been implemented in this driver.]

One common TCP/IP stack configuration to use with 802.11b in an access-point environment might be the the DHCP configuration, which is described in the ***TCP/IP User's Manual***. The following source code provides an example of a configuration that will allow joining an open access point that has DHCP enabled.

```
-----------------------------
#define TCPCONFIG 5
#define DISABLE_ETHERNET_AUTOCONF
#define PKTDRV "cfprism.lib"
#use dcrtcp.lib
main(){
     sock_init();
     while(1){
          tcp_tick(NULL);
     }
}
-----------------------------
```

This program will join any access point that is open, and will then attempt to discover its IP, netmask, gateway, and DNS address information. Now you can try to ping the RCM3100 with the ComplactFlash card from another host.

Note that you can discover what IP address was assigned by accessing the administrative function of your access point. Most access points allow Web-based configuration and access to status information, including DHCP leases that are active. Check your documentation for your access point documentation.

Compile-Time Configuration

The default 802.11b configuration that the driver will use is as follows.

Mode : infrastructure

SSID : "any"

WEP : off

You can change this configuration by setting compile-time options. A special **`define WIFI_INIT`** is a list of C instructions that will be executed when the interface is first brought up. You can modify the **`SCAN.C`** sample program that you ran with the ***Getting Started*** instructions to only join a specific access point by specifying an specific SSID as follows.

```
-------------------------------------
#define TCPCONFIG 5
#define DISABLE_ETHERNET_AUTOCONF
#define PKTDRV "cfprism.lib"
#define WIFI_INIT \
     wifi_ioctl(NULL,WIFI_MODE,"BSS",0);\
     wifi_ioctl(NULL,WIFI_SSID,"YourAccessPoint",0);

#use dcrtcp.lib
main(){
     sock_init();
     while(1){
          tcp_tick(NULL);
     }
}
-------------------------------------
```

This code forces the card into the following configuration.

Mode : infrastructure

SSID : "YourAccessPoint"

WEP : off

The card will only join an access point that has its SSID set to "YourAccessPoint". These other calls can also be used at compile time.

```
wifi_ioctl(NULL,WIFI_MODE,"IBBS",0);       // ad-hoc mode
wifi_ioctl(NULL,WIFI_OWNCHAN,"2",0);       // set channel
wifi_ioctl(NULL,WIFI_OWNSSID,"MySSID",0); // set SSID for ad-hoc
```

WEP encryption is handled by the interface card's firmware and thus does not burden the processor. Z-World recommends that you use WEP encryption whenever possible. The CompactFlash card used in the 802.11b (Wi-Fi) Application Kit kit supports 64-/128-bit WEP encryption.

Use the following configuration to use WEP encryption.

```
...
unsigned key0[13] = { 0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56 };
unsigned key1[13] = { 0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56 };
unsigned key2[13] = { 0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56 };
unsigned key3[13] = { 0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56,0x78,0x90,0x12,0x34,0x56 };

wifi_ioctl(NULL,WIFI_WEP_KEY0,key0,sizeof(key0)); // set 128bit (13 bytes) key
wifi_ioctl(NULL,WIFI_WEP_KEY1,key1,sizeof(key1)); // set 128bit (13 bytes) key
wifi_ioctl(NULL,WIFI_WEP_KEY2,key2,sizeof(key2)); // set 128bit (13 bytes) key
wifi_ioctl(NULL,WIFI_WEP_KEY3,key3,sizeof(key3)); // set 128bit (13 bytes) key
wifi_ioctl(NULL,WIFI_WEP_USEKEY,"0",0);           // use KEY0 for xmit
wifi_ioctl(NULL,WIFI_WEP_AUTH,"1",0);             // private authentication
wifi_ioctl(NULL,WIFI_WEP_FLAG,"1",0);              // enable WEP
```

The keys have to match in all of the devices that want to use the same access point and communicate with each other.

Run-Time Configuration

All the **`wifi_ioctl()`** calls used in the compile time configuration, which are defined via **`#define WIFI_INIT`**, can also be used at run-time. To change the configuration of the card, use the following calls to turn off the MAC before you changing the configuration information.

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

Then make the following call to enable the card when the configuration has been set.

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

Additional functions are also available, and are used in the **`SCAN.C`** sample program.

```
wifi_ioctl(NULL,WIFI_ROAM,"auto");   // "auto", "host" and "manual"
```

The above function will alter the card’s roaming behavior.

```
wifi_ioctl(NULL,WIFI_SCANREQ,"",0);                     // do a scan
wifi_ioctl(NULL,WIFI_SCANRES,scanbuf,sizeof(scanbuf)); // read result
```

The above functions will request the card to perform a scan and read back the result from the scan request. The card has to be in a “manual” roam mode for this function to work properly.

Functions

There is one basic API in the **`CFPRISM.LIB`** library.

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

PARAMETERS

`NULL` pointer

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

`cmd`	`*data`	`len`	Description
`WIFI_SSID`	`char*`	`0`	Set SSID string
`WIFI_MODE`	`char*`	`0`	"IBBS" or "BSS"
`WIFI_OWNCHAN`	`char*`	`0`	"0–14"
`WIFI_OWNSSID`	`char*`	`0`	"0–14"
`WIFI_WEP_FLAG`	`char*`	`0`	"0" or "1"
`WIFI_WEP_USEKEY`	`char*`	`0`	"0" through "3"
`WIFI_WEP_KEY0`	`char[]`	`5,13`	64-/128-bit key
`WIFI_WEP_KEY1`	`char[]`	`5,13`	64-/128-bit key
`WIFI_WEP_KEY2`	`char[]`	`5,13`	64-/128-bit key
`WIFI_WEP_KEY3`	`char[]`	`5,13`	64-/128-bit key
`WIFI_WEP_AUTH`	`char*`	`0`	"0" or "1" ("" = off)
`WIFI_STATUS`	`struct wifi_status*`	`sizeof(struct wifi_status)`	
`WIFI_SCANRES`	`struct wifi_scanres*`	`sizeof(struct wifi_scanres)`	
`WIFI_MAC`	`char*`	`0`	"on" "off"
`WIFI_SCANREQ`	`NULL`	`0`	
`WIFI_ROAM`	`char*`	`0`	"auto", "host", "manual"

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 WEP encryption enabled, do not execute any of the WEP commands in the table.

Sample Programs

Sample programs are provided in the Dynamic C `Samples\WiFi` folder. Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

Other directories in the `Samples` folder contain specific sample programs that illustrate the use of the corresponding Dynamic C libraries.

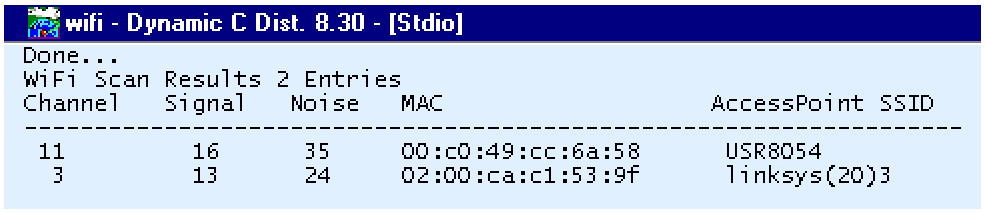
To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The RCM3100 module must be in the **Program** mode (which it will be when the programming cable is connected) and must be connected to a PC using the programming cable as described in the ***802.11b (Wi-Fi) Application Kit Getting Started*** instructions.

SCAN.C

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

Once the scan request has been issued, the host RCM3100 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....Done`**, and will display a list of access points/ad-hoc hosts as shown here.



```
Done...
WiFi Scan Results 2 Entries
Channel    Signal    Noise    MAC                          AccessPoint SSID
---------------------------------------------------------------------------
 11          16       35      00:c0:49:cc:6a:58             USR8054
  3          13       24      02:00:ca:c1:53:9f             linksys(20)3
```

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

- Channel—the channel the access point is on (1–11).
- Signal—the signal strength of the access point.
- Noise—the average noise of the channel.
- MAC—the hardware (MAC) address of access point.
- Access Point SSID—the SSID the access point is using.

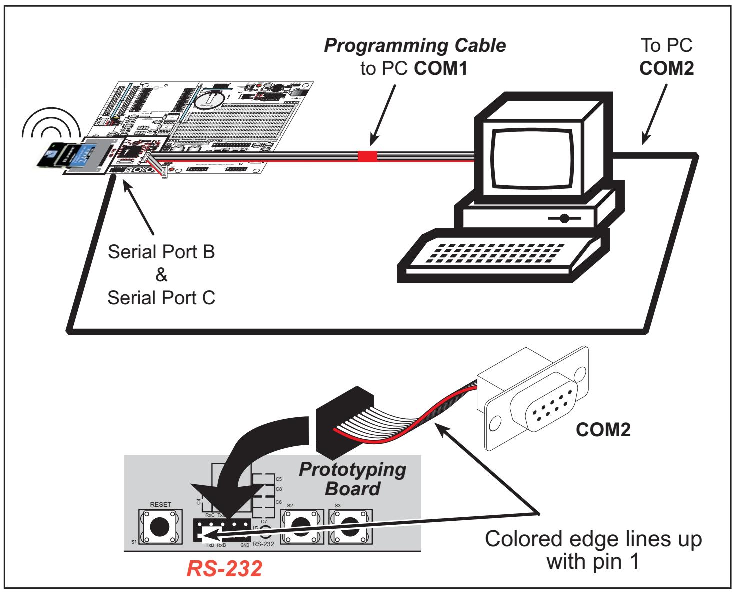
The LEDs on the Prototyping Board indicate the number of stations found.

LED DS1	LED DS2	No. of Stations Found
OFF	OFF	0
OFF	ON	1
ON	OFF	2
ON	ON	3 or more

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 the two asynchronous serial ports available on the Prototyping Board.

To run this sample program, you will need a separate PDA/laptop with 802.11b wireless compatibility. To begin, connect Serial Port B on the Prototyping Board to an available COM port on your PC using the serial cable as shown in the diagram. If your PC only has one COM port (or one USB port), compile and run **`WIFISERIAL.C`** first with the programming cable attached, then connect Serial Port B.



Programming Cable to PC COM1
To PC COM2
Serial Port B & Serial Port C
COM2
Prototyping Board
RESET
RS-232
Colored edge lines up with pin 1

Serial Port Connections

As shipped, the serial port configuration is as follows.

Serial Port B (TCP port 3023, 57600 baud)

Serial Port C (TCP port 23, 57600 baud)

The Wi-Fi configuration for this program is as follows.

Mode : infrastructure

SSID : "any"

The TCP/IP configuration is

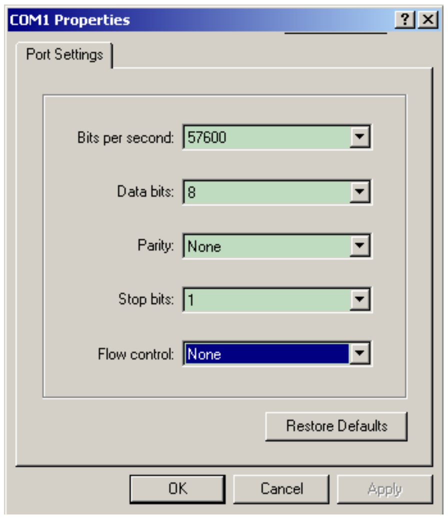
```
TCPCONFIG 1
```

which is the static IP configuration

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

The TCP/IP configuration is set via the **`TCPCONFIG`** macro in the **`TCP_CONFIG.LIB`** library. Most Z-World sample programs use **`TCPCONFIG 1`**. See the **`TCP_CONFIG.LIB`** library in the Dynamic C **`Lib\TCPIP`** directory for more information about other macro definitions.

Open HyperTerminal on your PC and set the properties for a COM1 or COM2 connection, depending on which PC serial port is connected to Serial Port B.


COM1 Properties

Port Settings

Bits per second: 57600

Data bits: 8

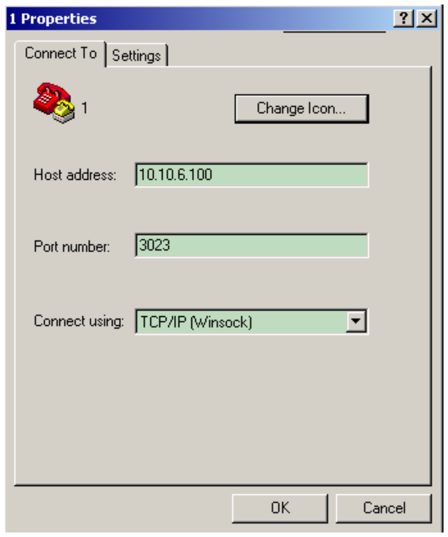
Parity: None

Stop bits: 1

Flow control: None

Next, open HyperTerminal on the wireless PDA/laptop, and set the IP address and port to match the **`_PRIMARY_STATIC_IP`** address and port number specified above for Serial Port B.

You may now type away in one HyperTerminal window, and your message will be displayed in the other HyperTerminal window. Note that there is no character echo, so you will not be able to view the text you are typing in the HyperTerminal window where you are typing.


1 Properties

Connect To | Settings

1

Host address: 10.10.6.100

Port number: 3023

Connect using: TCP/IP (Winsock)

DATALOGGER.C

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

The configuration required to connect to this sample program in its shipped configuration is as follows:

Mode : infrastructure

SSID : "any"

The TCP/IP configuration is

```
TCPCONFIG 1
```

which is the static IP configuration

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

The TCP/IP configuration is set via the **`TCPCONFIG`** macro in the **`TCP_CONFIG.LIB`** library. Most Z-World sample programs use **`TCPCONFIG 1`**. See the **`TCP_CONFIG.LIB`** library in the Dynamic C **`Lib\TCPIP`** directory for more information about other macro definitions.

The **`DATALOGGER.C`** sample program is designed for use with an Analog Devices TMP04F temperature sensor. Connect the temperature sensor to the Prototyping Board as follows.

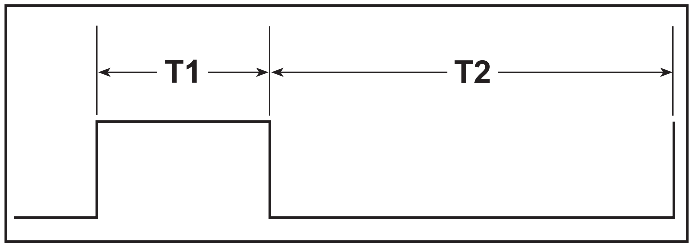
- V+ (2) to +5 V
- GND (3) to GND
- D_{OUT} [1] to PF7

This temperature sensor outputs PWM signals that can be used to compute the temperature.

The figure at right shows the timing diagram with T1 = high period and T2 = low period.

$$\text{Temperature (°C)} = 235 - \left(\frac{400 \times \text{T1}}{\text{T2}}\right)$$

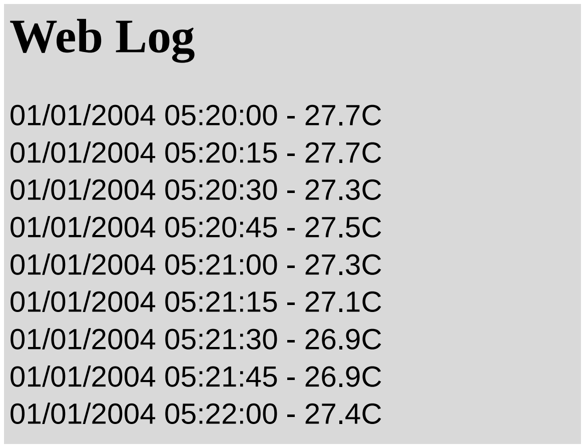
$$\text{Temperature (°F)} = 720 - \left(\frac{720 \times \text{T1}}{\text{T2}}\right)$$



Analog Devices TMP04F Output Format

The Rabbit 3000 microprocessor has pulse width capture hardware that can measure pulse timings. We use both channels on the same pin, but measure both the high and the low periods into the variables **`tmp04_T1`** and **`tmp04_T2`**.

The output from this sample program in your Web browser will resemble the following sample output. Simply open the Web browser using the IP address identified in the **`_PRIMARY_STATIC_IP`** line.



Web Log

```
01/01/2004 05:20:00 - 27.7C
01/01/2004 05:20:15 - 27.7C
01/01/2004 05:20:30 - 27.3C
01/01/2004 05:20:45 - 27.5C
01/01/2004 05:21:00 - 27.3C
01/01/2004 05:21:15 - 27.1C
01/01/2004 05:21:30 - 26.9C
01/01/2004 05:21:45 - 26.9C
01/01/2004 05:22:00 - 27.4C
```

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. The sample program allows pinging from a remote host on the network.

To run this sample program, you will need a separate PDA/laptop with 802.11b wireless compatibility. Once you're ready, just compile and run the sample program. The Dynamic C **STDIO** window will display any activity on the host PC connected to the RCM3100.

The Wi-Fi configuration for this program is as follows.

Mode : infrastructure

SSID : "any"

The TCP/IP configuration is

```
TCPCONFIG 1
```

which is the static IP configuration

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

The TCP/IP configuration is set via the **`TCPCONFIG`** macro in the **`TCP_CONFIG.LIB`** library. Most Z-World sample programs use **`TCPCONFIG 1`**. See the **`TCP_CONFIG.LIB`** library in the Dynamic C **`Lib\TCPIP`** directory for more information about other macro definitions.

Next, open a command window on the wireless PDA/laptop, and type the command

```
ping IP "<PRIMARY_STATIC_IP>"
```

where **`"<PRIMARY_STATIC_IP>"`** is the IP address set in the above configuration. One or more reply lines will then appear to display the results of the ping.

Helpful Hints

1. When an access point is turned off after a CompactFlash card has joined it, the Linksys WCF12 card will remain in the “joined” state and will wait for the access point 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 RCM3100, and you change the wireless 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
   ```

   Wireless 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 wireless CompactFlash cards. The interface and software driver are not designed to allow hot-swapping.

4. The CompactFlash Adapter Board in this kit was designed to be used with Wi-Fi CompactFlash cards. It has not been tested for compatibility with other types of CompactFlash cards.

5. When using the `wifi_ioctl()` calls, a call with a zero-length string `""` will cause the requested operation to not execute.

   ```
   wifi_ioctl(NULL,WIFI_OWNCHAN,"",0); // don't execute this command
   ```

   Use the following command if you want the operation to execute.

   ```
   wifi_ioctl(NULL,WIFI_OWNCHAN,"0",0); // set the channel to the default
   ```

   In particular, you should use the following lines if you do not want to enable WEP.

   ```
   wifi_ioctl(NULL,WIFI_WEP_FLAG,"",0);
   wifi_ioctl(NULL,WIFI_WEP_AUTH,"",0);
   ```

   Better yet, just don’t use them at all. The only exception to this is SSID where a zero-length string will set the SSID to "none" or "any".

6. Check www.zworld.com/products/WiFi_App_Kit/ for the latest information on the 802.11b (Wi-Fi) Application Kit. The latest software updates are available at www.zworld.com/support/downloads/downloads_prod.shtml.

CompactFlash Cards

The following CompactFlash cards were evaluated by Z-World for the 802.11b (Wi-Fi) Application Kit, and were found to provide satisfactory results.

Manufacturer and Model	Hardware	Onboard Firmware	Comments
Ambicom WL1100C-CF	—	—	
Belkin F5D6060	—	—	Type 1.5
Compex iWavePort WCF11	—	—	
D-Link DCF-660W	v. A1	1.3.6	
Hawking Technology H-CF30W	—	—	
Linksys WCF11	—	—	Type 1.5
Linksys WCF12	—	—	
Netgear MA701	v. 1.0	1.4.9	
Proxim ORiNOCO Classic Gold PC Card	—	—	Needs CompactFlash to PCMCIA adapter, supports external antenna
SanDisk SDWCFB-000-768	v 1.0	1.4.9	
SMC SMC2642W	—	1.3.6	
TRENDnet TEW-222CF	—	1.4.2	

References

Matthew S. Gast, ***802.11 Wireless Networking***; O'Reilly Media, Incorporated, 2002.

TCP/IP User's Manual; Z-World Online Documentation.