Digi - 20-101-1246 Datasheet

Welcome to E-XFL.COM

Understanding Embedded - Microcontroller, Microprocessor, FPGA Modules

Embedded - Microcontroller, Microprocessor, and FPGA Modules are fundamental components in modern electronic systems, offering a wide range of functionalities and capabilities. Microcontrollers are compact integrated circuits designed to execute specific control tasks within an embedded system. They typically include a processor, memory, and input/output peripherals on a single chip. Microprocessors, on the other hand, are more powerful processing units used in complex computing tasks, often requiring external memory and peripherals. FPGAs (Field Programmable Gate Arrays) are highly flexible devices that can be configured by the user to perform specific logic functions, making them invaluable in applications requiring customization and adaptability.

Applications of Embedded - Microcontroller,

Details	
Product Status	Obsolete
Module/Board Type	MPU Core
Core Processor	Rabbit 5000
Co-Processor	-
Speed	73.73MHz
Flash Size	1.5MB
RAM Size	1MB
Connector Type	IDC Header 2x25, 2x5, 1xAntenna
Size / Dimension	1.84" x 2.85" (47mm x 72mm)
Operating Temperature	-30°C ~ 75°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/20-101-1246

The RCM5400W series is programmed over a standard PC USB port through a programming cable supplied with the Development Kit. The RCM5400W may also be programmed remotely using the Remote Program Update library with Dynamic C v. 10.54 or later. See Application Note AN421, ***Remote Program Update***, for more information.

NOTE: The RabbitLink cannot be used to program the RCM5400W.

Appendix A provides detailed specifications for the RCM5400W.

1.2 Advantages of the RCM5400W

- Fast time to market using a fully engineered, "ready-to-run/ready-to-program" microprocessor core module.
- Competitive pricing when compared with the alternative of purchasing and assembling individual components.
- Easy C-language program development and debugging
- Rabbit Field Utility to download compiled Dynamic C .bin files, and cloning board options for rapid production loading of programs.
- Generous memory size allows large programs with tens of thousands of lines of code, and substantial data storage.
- Easily scalable for commercial deployment applications

2.2.3 Step 3 — Attach Module to Prototyping Board

Turn the RCM5400W module so that the mounting holes line up with the corresponding holes on the Prototyping Board. Insert the metal standoffs as shown in Figure 4, secure them from the bottom using the 4-40 × 3/16 screws and washers, then insert the module's header J1 on the bottom side into socket RCM1 on the Prototyping Board.

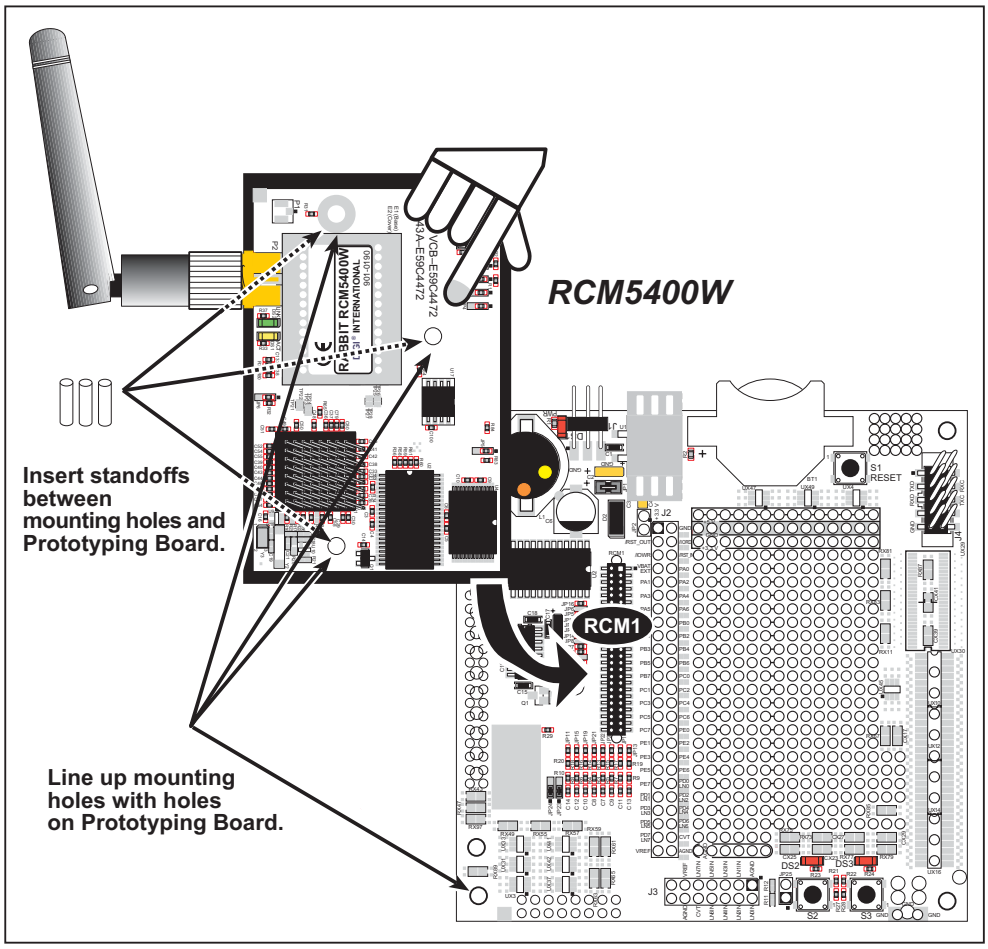

Figure 4. Install the Module on the Prototyping Board

NOTE: It is important that you line up the pins on header J1 of the module exactly with socket RCM1 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work. Permanent electrical damage to the module may also result if a misaligned module is powered up.

Press the module's pins gently into the Prototyping Board socket—press down in the area above the header pins. For additional integrity, you may secure the RCM5400W to the standoffs from the top using the remaining three screws and washers.

3.2.1 Use of Serial Flash

The following sample programs can be found in the **SAMPLES\RCM5400W\Serial_Flash** folder.

- **SERIAL_FLASHLOG.C**—This program runs a simple Web server and stores a log of hits on the home page of the serial flash "server." This log can be viewed and cleared from a browser at http://10.10.6.100/. You will likely have to first "configure" your network interface card for a "10Base-T Half-Duplex," "100Base-T Half-Duplex," or an "Auto-Negotiation" connection on the "Advanced" tab, which is accessed from the control panel (**Start > Settings > Control Panel**) by choosing **Network Connections**.
- **SFLASH_INSPECT.C**—This program is a handy utility for inspecting the contents of a serial flash chip. When the sample program starts running, it attempts to initialize a serial flash chip on Serial Port B. Once a serial flash chip is found, the user can perform five different commands to print out the contents of a specified page, set all bytes on the specified page to a single random value, clear (set to zero) all the bytes in a specified page, set all bytes on the specified page to a given value, or save user-specified text to a selected page.

4.1 RCM5400W Digital Inputs and Outputs

Figure 7 shows the RCM5400W pinouts for header J1.

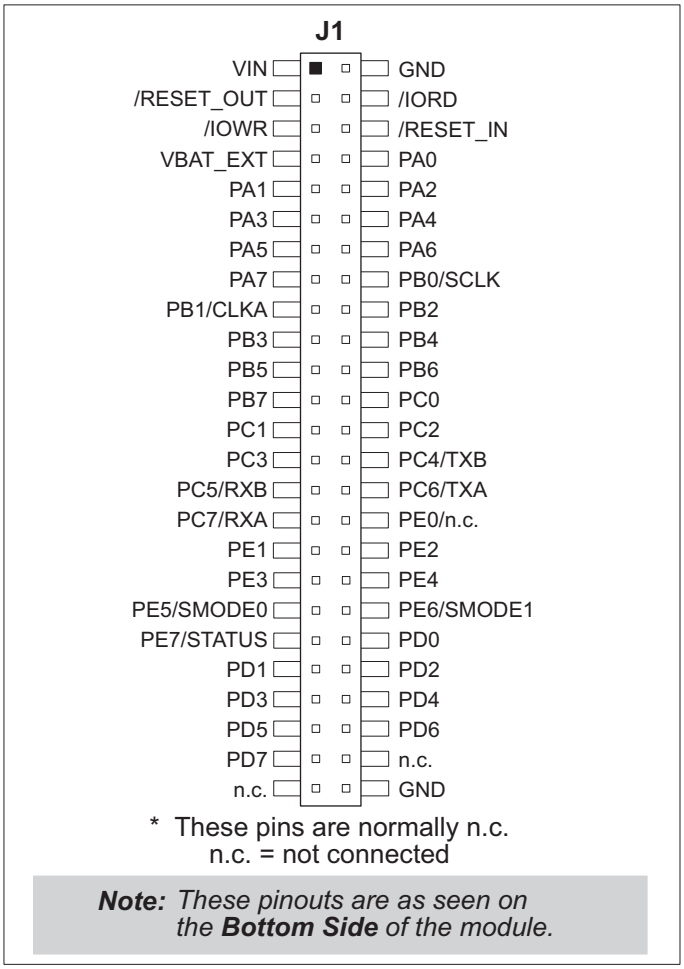

Figure 7. RCM5400W Pinout

Headers J1 is a standard 2 × 25 IDC header with a nominal 1.27 mm pitch.

4.5 Other Hardware

4.5.1 Clock Doubler

The RCM5400W takes advantage of the Rabbit 5000 microprocessor's internal clock doubler. A built-in clock doubler allows half-frequency crystals to be used to reduce radiated emissions. The 73.73 MHz frequency specified for the RCM5400W is generated using a 36.864 MHz crystal.

The clock doubler should not be disabled since Wi-Fi operations depend highly on CPU resources.

4.5.2 Spectrum Spreader

The Rabbit 5000 features a spectrum spreader, which helps to mitigate EMI problems. The spectrum spreader is on by default, but it may also be turned off or set to a stronger setting. The spectrum spreader settings may be changed through a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Normal spreading is the default, and usually no entry is needed. If you need to specify normal spreading, add the line

   ```
   ENABLE_SPREADER=1
   ```

   For strong spreading, add the line

   ```
   ENABLE_SPREADER=2
   ```

   To disable the spectrum spreader, add the line

   ```
   ENABLE_SPREADER=0
   ```

   NOTE: The strong spectrum-spreading setting is not recommended since it may limit the maximum clock speed or the maximum baud rate. It is unlikely that the strong setting will be needed in a real application.

3. Click **OK** to save the macro. The spectrum spreader will now remain off whenever you are in the project file where you defined the macro.

NOTE: Refer to the ***Rabbit 5000 Microprocessor User's Manual*** for more information on the spectrum-spreading setting and the maximum clock speed.

5. SOFTWARE REFERENCE

Dynamic C is an integrated development system for writing embedded software. It runs on a Windows-based PC and is designed for use with single-board computers and other devices based on the Rabbit microprocessor. Chapter 5 describes the libraries and function calls related to the RCM5400W.

5.1 More About Dynamic C

Dynamic C has been in use worldwide since 1989. It is specially designed for programming embedded systems, and features quick compile and interactive debugging. A complete reference guide to Dynamic C is contained in the ***Dynamic C User's Manual***.

You have a choice of doing your software development in the flash memory or in the static SRAM included on the RCM5400W. The flash memory and SRAM options are selected with the **Options > Program Options > Compiler** menu.

The advantage of working in RAM is to save wear on the flash memory, which is limited to about 100,000 write cycles. The disadvantage is that the code and data might not both fit in RAM.

> **NOTE:** Do not depend on the flash memory sector size or type in your program logic. The RCM5400W and Dynamic C were designed to accommodate flash devices with various sector sizes in response to the volatility of the flash-memory market.

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows NT and later—see Rabbit's Technical Note TN257, *Running Dynamic C® With Windows Vista®*, for additional information if you are using a Dynamic C under Windows Vista. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

5.2.6 Prototyping Board Function Calls

The function calls described in this section are for use with the Prototyping Board features. The source code is in the Dynamic C **LIB\Rabbit4000\RCM5xxx\RCM54xxW.LIB** library if you need to modify it for your own board design.

The sample programs in the Dynamic C **SAMPLES\RCM5400W** folder illustrate the use of the function calls.

Other generic functions applicable to all devices based on Rabbit microprocessors are described in the ***Dynamic C Function Reference Manual***.

5.2.6.1 Board Initialization

---

brdInit

```
void brdInit(void);
```

DESCRIPTION

Call this function at the beginning of your program. This function initializes Parallel Ports A through E for use with the Prototyping Board. This function call is intended for demonstration purposes only, and can be modified for your applications.

Summary of Initialization

1. I/O port pins are configured for Prototyping Board operation.
2. Unused configurable I/O are set as tied outputs.
3. RS-232 is not enabled.
4. LEDs are off.
5. The slave port is disabled.

RETURN VALUE

None.

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.

6.1.2 Ad-Hoc Mode

In the ad-hoc mode, each device can set a channel number and an 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.

6.1.3 Additional Information

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

6.2.3 Configuration Information

6.2.3.1 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 configure the RCM5400W modules with a default **TCPCONFIG** macro from the **LIB\Rabbit4000\TCPIP\TCP_CONFIG.LIB** library. This macro allows specific IP address, netmask, gateway, and Wi-Fi parameters to be set at compile time. Change the network settings to configure your RCM5400W module 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 as explained in Section 6.3.1.

The Wi-Fi configurations are contained within **TCPCONFIG 1** (no DHCP) and **TCPCONFIG 5** (with DHCP, used primarily with infrastructure mode). You will need to **#define TCPCONFIG 1** or **#define TCPCONFIG 5** at the beginning of your program.

> **NOTE:** **TCPCONFIG 0** is not supported for Wi-Fi applications.

There are some other "standard" configurations for **TCPCONFIG**. Their values are documented in the **LIB\Rabbit4000\TCPIP\TCP_CONFIG.LIB** library. More information is available in the ***Dynamic C TCP/IP User's Manual***.

- **WIFISCAN.C**—initializes the RCM5400W 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 RCM5400W does not actually join an 802.11 network. This program outputs the results of the scan to the Dynamic C **STDIO** window.
- **WIFISCANASSOCIATE.C**— demostrates how to scan Wi-Fi channels for SSIDs using **ifconfig IFS_WIFI_SCAN**. This takes a while to complete, so **ifconfig()** calls a callback function when it is done. The callback function is specified using **ifconfig IFS_WIFI_SCAN**.

  Before you run this sample program, configure the Dynamic C **TCP_CONFIG.LIB** library and your **TCPCONFIG** macro.

  1. Use macro definitions in the "Defines" tab in the Dynamic C **Options > Project Options** menu to modify any parameter settings.

     If you are not using DHCP, set the IP parameters to values appropriate to your network.

     ```
     _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"
     ```

     Set the macro **IFC_WIFI_SSID=** to define a C-style string to set the SSID of your access point as, for example,

     ```
     IFC_WIFI_SSID = "My Access Point"
     ```

     or use an empty string, **""**, to associate with the strongest BSS available.

     Alternatively, you may create your own **CUSTOM_CONFIG.LIB** library modeled on the Dynamic C **TCP_CONFIG.LIB** library. Then use a **TCPCONFIG** macro greater than or equal to 100, which will invoke your **CUSTOM_CONFIG.LIB** library to be used. Remember to add the **CUSTOM_CONFIG.LIB** library to **LIB.DIR**.
  2. If you are using DHCP, change the definition of the **TCPCONFIG** macro to 5. The default value of 1 indicates Wi-Fi with a static IP address.

  Now compile and run the sample program. Follow the menu options displayed in the Dynamic C **STDIO** window.

  ```
  Press s to scan available access points
  Press a to scan access points and associate
  Press m to print WIFI MAC status
  ```

  Note that **ifconfig IFS_WIFI_SCAN** function calls do not return data directly since the scan takes a fair amount of time. Instead, callback functions are used. The callback function is passed to **ifconfig()** as the only parameter to **IFS_WIFI_SCAN**.

  ```
  ifconfig(IF_WIFI0, IFS_WIFI_SCAN, scan_callback, IFS_END);
  ```

- **PINGLED_WPA_PSK.C**—This program demonstrates the use of WPA PSK (Wi-Fi Protected Access with Pre-Shared Key). WPA is a more secure replacement for WEP. The implementation in the sample program supports use of the TKIP (Temporal Key Integrity Protocol) cypher suite.

  The sample program uses macros to configure the access point for WPA PSK, specify the TKIP cypher suite, assign the access point SSID, and set the passphrase.

```
#define WIFI_USE_WPA  // Bring in WPA support
#define IFC_WIFI_ENCRYPTION IFPARAM_WIFI_ENCR_TKIP // Define cypher suite

#define IFC_WIFI_SSID "parvati"

#define IFC_WIFI_WPA_PSK_PASSPHRASE "now is the time"
```

  The next macro specifies a suitable pre-shared key to use instead of the passphrase. The key may be entered either as 64 hexadecimal digits or as an ASCII string of up to 63 characters.

```
#define IFC_WIFI_WPA_PSK_HEXSTR
```

  TIP: There is a good chance of typos since the key is long. First, enter the key in this sample program macro, then copy and paste it to your access point. This ensures that both the RCM5400W and the access point have the same key.

  TIP: For an initial test, it may be easier to use the 64 hex digit form of the key rather than the ASCII passphrase. A passphrase requires considerable computation effort, which delays the startup of the sample program by about 30 seconds.

  Change **PING_WHO** to the host you want to ping. You may modify **PING_DELAY** to change the amount of time in milliseconds between the outgoing pings.

  Uncomment the **VERBOSE** define to see the incoming ping replies.

  Once you have compiled the sample program and it is running, LED DS2 will flash when a ping is sent, and LED DS3 will flash when a ping is received.

These macros specify the WEP keys to use for WEP encryption. These keys can be either 40-bit or 104-bit (i.e., 5 bytes or 13 bytes). They must be defined as a comma-separated list of byte values.

Note that you do not necessarily need to define all four WEP keys. You may typically just define one key, but make sure it matches the key used on all other devices, and set **IFC_WIFI_WEP_KEYNUM** to point to the correct key.

If both **IFC_WIFI_WEP_KEY#_BIN** and **IFC_WIFI_WEP_KEY#_HEXSTR** are defined for a particular key, the hex version will be used.

- Use WPA encryption.

  The following macro must also be used to compile WPA functionality into the Wi-Fi driver. This is necessary to enable TKIP encryption.

  ```
  #define WIFI_USE_WPA
  ```

- Set WPA passphrase—**IFC_WIFI_WPA_PSK_PASSPHRASE** is a string that matches the passphrase on your access point. It may also point to a variable.

  Define an ASCII passphrase here, from 1 to 63 characters long. An example is shown below.

  ```
  #define IFC_WIFI_WPA_PSK_PASSPHRASE "now is the time"
  ```

  If possible, you should use **IFC_WIFI_WPA_PSK_HEXSTR** instead of **IFC_WIFI_WPA_PSK_PASSPHRASE** to set the key.

- Set WPA hexadecimal key—**IFC_WIFI_WPA_PSK_HEXSTR** is a string of hexadecimal digits that matches the 256-bit (64-byte) hexadecimal key used by your access point.

  Specify a 64 hexadecimal digit (256 bits) key here. This key will be used and will override any passphrase set with the **IFC_WIFI_WPA_PSK_PASSPHRASE** macro. The example hex key shown below

  ```
  #define IFC_WIFI_WPA_PSK_HEXSTR \
     "57A12204B7B350C4A86A507A8AF23C0E81D0319F4C4C4AE83CE3299EFE1FCD27"
  ```

  is valid for the SSID **"rabbitTest"** and the passphrase **"now is the time"**.

  Using a passphrase is rather slow. It takes a Rabbit 5000 more than 20 seconds to generate the actual 256-bit key from the passphrase. If you use a passphrase and **#define WIFI_VERBOSE_PASSPHRASE**, the Wi-Fi library will helpfully print out the hex key corresponding to that passphrase and SSID.

- Authentication algorithm—**IFC_WIFI_AUTHENTICATION** can be used to specify the authentication modes used.

  The default shown below allows enables both open-system authentication and shared-key authentication.

  ```
  #define IFPARAM_WIFI_AUTH_ANY
  ```

The following authentication options are available.

- **IFPARAM_WIFI_AUTH_OPEN** — only use open authentication.
- **IFPARAM_WIFI_AUTH_SHAREDKEY** — only use shared-key authentication (useful for WEP only).
- **IFPARAM_WIFI_WPA_PSK** — use WPA preshared-key authentication (useful for TKIP and CCMP only).

- Fragmentation threshold—**IFC_WIFI_FRAG_THRESHOLD** sets the fragmentation threshold. Frames (or packets) that are larger than this threshold are split into multiple fragments. This can be useful on busy or noisy networks. The value can be between **256** and **2346**.

  The default, **0**, means no fragmentation.

  ```
  #define IFC_WIFI_FRAG_THRESHOLD 0
  ```

- RTS threshold—**IFC_WIFI_RTS_THRESHOLD** sets the RTS threshold, the frame size at which the RTS/CTS mechanism is used. This is sometimes useful on busy or noisy networks. Its range is **0** to **2347**.

  The default, **2347**, means no RTS/CTS.

  ```
  #define IFC_WIFI_RTS_THRESHOLD 2347
  ```

Examples are available within Dynamic C. Select "Function Lookup" from the **Help** menu, or press **<ctrl-H>**. Type "TCPCONFIG" in the Function Search field, and hit **<Enter>**. Scroll down to the section on "Wi-Fi Configuration." The ***Dynamic C TCP/IP User's Manual***.(Volume 1) provides additional information about these macros and Wi-Fi.

It is also possible to redefine any of the above parameters dynamically using the **ifconfig()** function call. Macros for alternative Wi-Fi configurations are provided with the **ifconfig()** function call, and may be used to change the above default macros or configurations.

6.4 Where Do I Go From Here?

NOTE: If you purchased your RCM5400W or RCM5450W through a distributor or through a Rabbit 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 Technical Bulletin Board and forums at www.rabbit.com/support/bb/ and at www.rabbit,com/forums/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

If the sample programs ran fine, you are now ready to go on.

An Introduction to TCP/IP and the ***Dynamic C TCP/IP User's Manual***.provide background and reference information on TCP/IP, and are available on the CD and on our Web site.

A.1 Electrical and Mechanical Characteristics

Figure A-1 shows the mechanical dimensions for the RCM5400W.

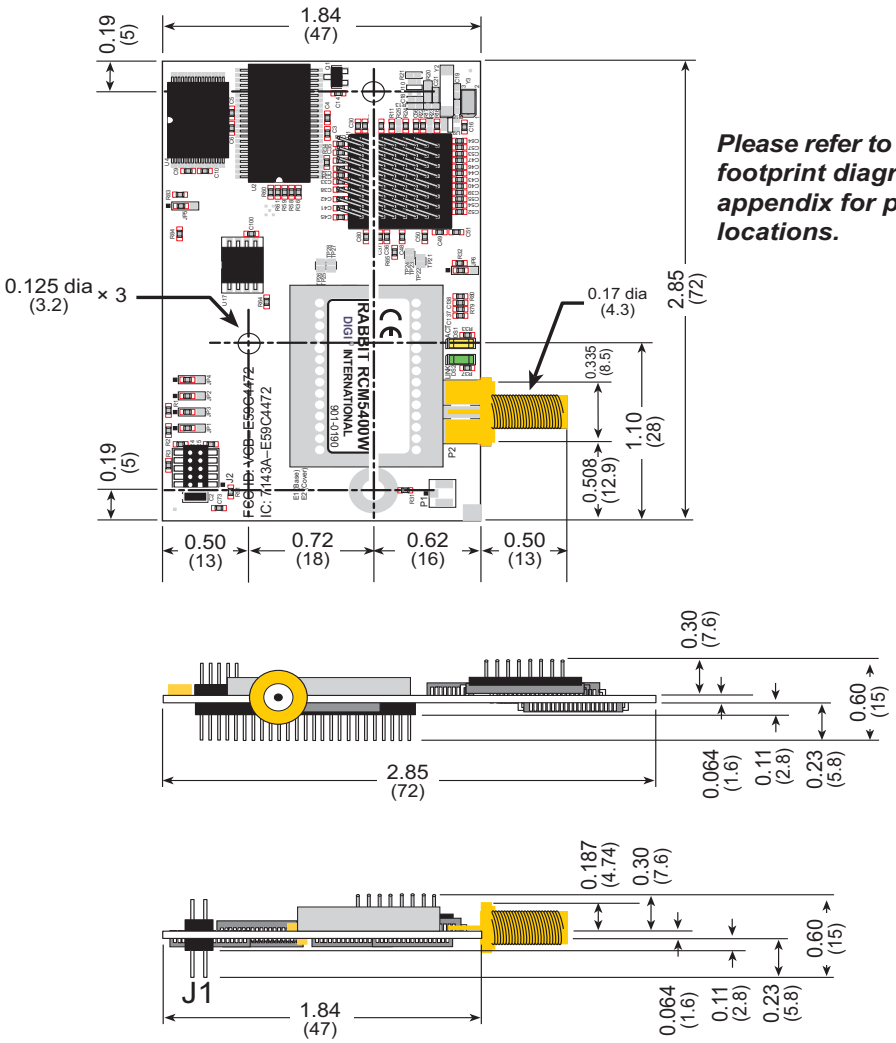

Figure A-1. RCM5400W Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses. All dimensions have a manufacturing tolerance of ±0.01" (0.25 mm).

It is recommended that you allow for an “exclusion zone” of 0.04" (1 mm) around the RCM5400W in all directions when the RCM5400W is incorporated into an assembly that includes other printed circuit boards. An “exclusion zone” of 0.08" (2 mm) is recommended below the RCM5400W when the RCM5400W is plugged into another assembly. Figure A-2 shows this “exclusion zone.”

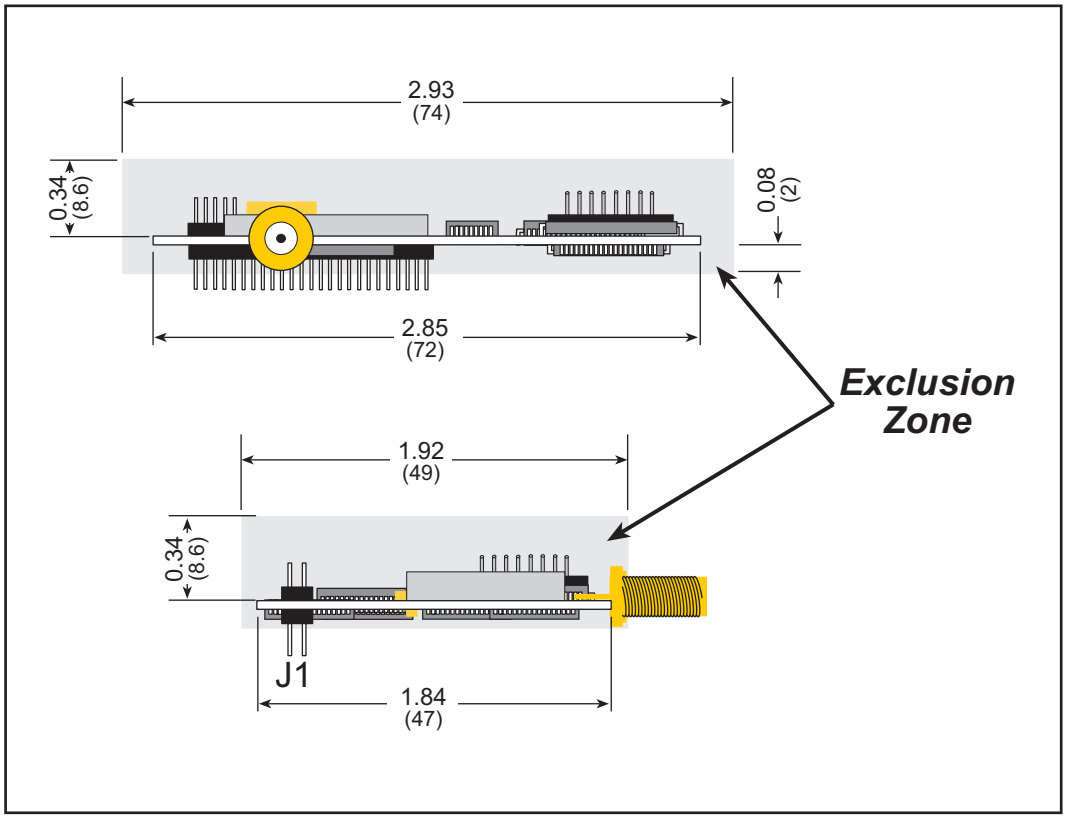

Figure A-2. RCM5400W “Exclusion Zone”

NOTE: There is an antenna associated with the RCM5400W RabbitCore modules. Do not use any RF-absorbing materials in these vicinities in order to realize the maximum range.

The RCM5400W modules were tested for heat dissipation over the specified operating temperature range, and normal heat dissipation by convection was found to be adequate. If you plan to use RCM5400W modules in a tightly enclosed space, additional forced-air cooling will likely be needed.

If you are planning to mount your RCM5400W directly in a panel-mounted enclosure, the RP-SMA antenna connector will extend outside the enclosure. Keep the thickness of the enclosure plus washer and lock nut to less than 0.2" (5 mm) to make sure that the antenna can be mounted securely in the RP-SMA antenna connector.

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM5400W.

Table A-1. RCM5400W Specifications

Parameter	RCM5400W	RCM5450W
Microprocessor	Rabbit® 5000 at 73.73 MHz	
Data SRAM	512K	512K
Program Execution Fast SRAM	512K	1MB
Flash Memory	512K	1MB
Serial Flash Memory	1MB	2MB
Backup Battery	Connection for user-supplied backup battery (to support RTC and data SRAM)	
General Purpose I/O	up to 40 parallel digital I/0 lines configurable with four layers of alternate functions	
Additional Inputs	Reset in	
Additional Outputs	Reset out	
External I/O Bus	Can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), plus I/O read/write	
Serial Ports	6 high-speed, CMOS-compatible ports: • all 6 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 2 as SDLC/HDLC • 1 asynchronous clocked serial port shared with programming port • 1 clocked serial port shared with serial flash	
Serial Rate	Maximum asynchronous baud rate = CLK/8	
Slave Interface	Slave port allows the RCM5400W to be used as an intelligent peripheral device slaved to a master processor	
Real Time Clock	Yes	
Timers	Ten 8-bit timers (6 cascadable from the first), one 10-bit timer with 2 match registers, and one 16-bit timer with 4 outputs and 8 set/reset registers	
Watchdog/Supervisor	Yes	
Pulse-Width Modulators	4 channels synchronized PWM with 10-bit counter 4 channels variable-phase or synchronized PWM with 16-bit counter	
Input Capture	2-channel input capture can be used to time input signals from various port pins	
Quadrature Decoder	2-channel quadrature decoder accepts inputs from external incremental encoder modules	

A.3 I/O Buffer Sourcing and Sinking Limit

Unless otherwise specified, the Rabbit I/O buffers are capable of sourcing and sinking 8 mA of current per pin at full AC switching speed. Full AC switching assumes a 100 MHz CPU clock with the clock doubler enabled and capacitive loading on address and data lines of less than 70 pF per pin. The absolute maximum operating voltage on all I/O is 3.3 V ±10%.

A.4 Bus Loading

You must pay careful attention to bus loading when designing an interface to the RCM5400W. This section provides bus loading information for external devices.

Table A-4 lists the capacitance for the various RCM5400W I/O ports.

Table A-4. Capacitance of Rabbit 5000 I/O Ports

I/O Ports	Input Capacitance (pF)	Output Capacitance (pF)
Parallel Ports A to E	12	14

Table A-5 lists the external capacitive bus loading for the various RCM5400W output ports. Be sure to add the loads for the devices you are using in your custom system and verify that they do not exceed the values in Table A-5.

Table A-5. External Capacitive Bus Loading -20°C to +85°C

Output Port	Clock Speed (MHz)	Maximum External Capacitive Loading (pF)
All I/O lines with clock doubler enabled	73.73	100

The drain on the battery by the RCM5400W is typically 7.5 µA when no other power is supplied. If a 165 mA·h battery is used, the battery can last about 2.5 years:

$$\frac{165\ \text{mA}\cdot\text{h}}{7.5\ \mu\text{A}} = 2.5\ \text{years}.$$

The actual battery life in your application will depend on the current drawn by components not on the RCM5400W and on the storage capacity of the battery. The RCM5400W does not drain the battery while it is powered up normally.

Cycle the main power off/on after you install a backup battery for the first time, and whenever you replace the battery. This step will minimize the current drawn by the real-time clock oscillator circuit from the backup battery should the RCM5400W experience a loss of main power.

NOTE: Remember to cycle the main power off/on any time the RCM5400W is removed from the Prototyping Board or motherboard since that is where the backup battery would be located.

Rabbit's Technical Note TN235, ***External 32.768 kHz Oscillator Circuits***, provides additional information about the current draw by the real-time clock oscillator circuit.

C.1.2 Battery-Backup Circuit

Figure C-2 shows the battery-backup circuit.

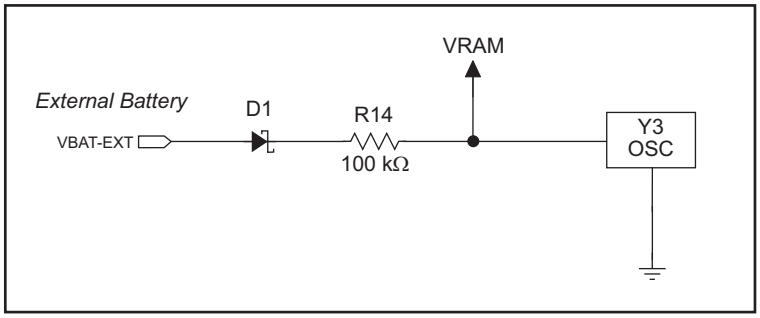

Figure C-2. RCM5400W Backup Battery Circuit

The battery-backup circuit serves three purposes:

- It reduces the battery voltage to the SRAM and to the real-time clock, thereby limiting the current consumed by the real-time clock and lengthening the battery life.
- It ensures that current can flow only ***out*** of the battery to prevent charging the battery.
- A voltage, VRAM, is supplied to Y3, the integrated 32.768 kHz oscillator, which keeps working when the voltage begins to drop.