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 3000
Co-Processor	-
Speed	44.2MHz
Flash Size	512KB (Internal), 8MB (External)
RAM Size	1MB
Connector Type	2 IDC Headers 2x17, 1 IDC Header 2x5
Size / Dimension	1.85" x 2.73" (47mm x 69mm)
Operating Temperature	-40°C ~ 70°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/101-1067

2.2.2 Step 2 — Connect Programming Cable

The programming cable connects the RCM3305 series module to the PC running Dynamic C to download programs and to monitor the module during debugging.

2.2.2.1 RCM3309 and RCM3319

Connect the 10-pin connector of the programming cable labeled **PROG** to header J1 on the RCM3309/RCM3319 as shown in Figure 3(a). There is a small dot on the circuit board next to pin 1 of header J1. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a non-programming serial connection.)

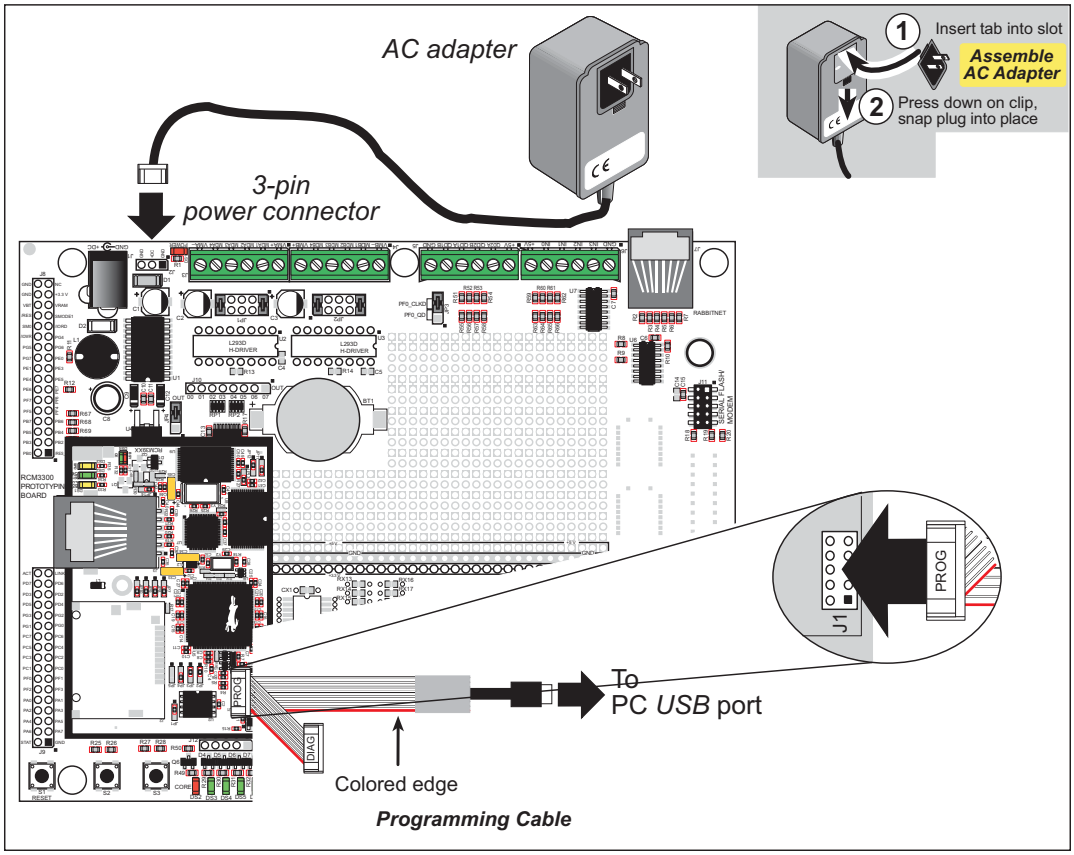


Figure 3(a). Connect Programming Cable and Power Supply

Connect the other end of the programming cable to an available USB port on your PC or workstation. Your PC should recognize the new USB hardware, and the LEDs in the shrink-wrapped area of the USB programming cable will flash.

3.2.1 Use of Serial Flash

3.2.1.1 Onboard Serial Flash

The following sample programs can be found in the **SAMPLES\RCM3300\SerialFlash** folder.

- **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 two different commands to either print out the contents of a specified page or clear (set to zero) all the bytes in a specified page.
- **SFLASH_LOG.C**—This program runs a simple Web server and stores a log of hits in the serial flash. This log can be viewed and cleared from a browser.

3.2.1.2 SF1000 Serial Flash Card

The following sample program can be found in the **SAMPLES\RCM3300\SF1000** folder.

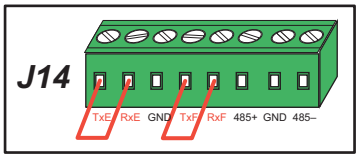
- **SERFLASHTEST.C**—An optional SF1000 Serial Flash card is required to run this demonstration. Install the Serial Flash card into socket J11 on the Prototyping Board. This sample program demonstrates how to read and write from/to the Serial Flash card.

3.2.2 Serial Communication

The following sample programs can be found in the **SAMPLES\RCM3300\SERIAL** folder.

- **FLOWCONTROL.C**—This program demonstrates hardware flow control by configuring Serial Port F for CTS/RTS with serial data coming from TxE (Serial Port E) at 115,200 bps. One character at a time is received and is displayed in the **STDIO** window.

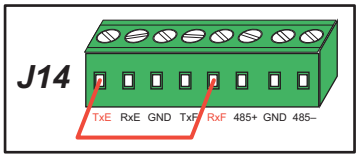
  To set up the Prototyping Board, you will need to tie TxE and RxE together on the RS-232 header at J14, and you will also tie TxF and RxF together as shown in the diagram.

  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of no flow control.

- **PARITY.C**—This program demonstrates the use of parity modes by repeatedly sending byte values 0–127 from Serial Port E to Serial Port F. The program will switch between generating parity or not on Serial Port E. Serial Port F will always be checking parity, so parity errors should occur during every other sequence.

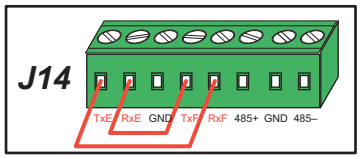
  To set up the Prototyping Board, you will need to tie TxE and RxF together on the RS-232 header at J14 as shown in the diagram.

  The Dynamic C **STDIO** window will display the error sequence.

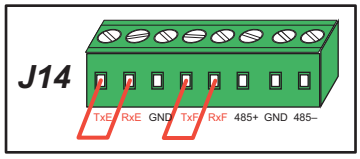
- **SIMPLE3WIRE.C**—This program demonstrates basic RS-232 serial communication. Lower case characters are sent by TxE, and are received by RxF. The characters are converted to upper case and are sent out by TxF, are received by RxE, and are displayed in the Dynamic C **STDIO** window.

  To set up the Prototyping Board, you will need to tie TxE and RxF together on the RS-232 header at J14, and you will also tie RxE and TxF together as shown in the diagram.



- **SIMPLE5WIRE.C**—This program demonstrates 5-wire RS-232 serial communication with flow control on Serial Port F and data flow on Serial Port E.

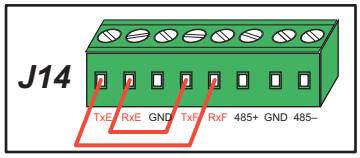
  To set up the Prototyping Board, you will need to tie TxE and RxE together on the RS-232 header at J14, and you will also tie TxF and RxF together as shown in the diagram.



  Once you have compiled and run this program, you can test flow control by disconnecting TxF from RxF while the program is running. Characters will no longer appear in the **STDIO** window, and will display again once TxF is connected back to RxF.

- **SWITCHCHAR.C**—This program transmits and then receives an ASCII string on Serial Ports E and F. It also displays the serial data received from both ports in the **STDIO** window.

  To set up the Prototyping Board, you will need to tie TxE and RxF together on the RS-232 header at J14, and you will also tie RxE and TxF together as shown in the diagram.



  Once you have compiled and run this program, press and release S2 and S3 on the Prototyping Board. The data sent between the serial ports will be displayed in the **STDIO** window.

Two sample programs, **SIMPLE485MASTER.C** and **SIMPLE485SLAVE.C**, are available to illustrate RS-485 master/slave communication. To run these sample programs, you will need a second Rabbit-based system with RS-485—another Rabbit single-board computer or RabbitCore module may be used as long as you use the master or slave sample program associated with that board.

Before running either of these sample programs on the RCM3305/RCM3315 assembly, make sure pins 1–2 and pins 5–6 are jumpered together on header JP5 to use the RS-485 bias and termination resistors. The sample programs use Serial Port C as the RS-485 serial port, and they use PD7 to enable/disable the RS-485 transmitter.

4.2 Serial Communication

The RCM3305/RCM3315 does not have any serial transceivers directly on the board. However, a serial interface may be incorporated into the board the RCM3305/RCM3315 is mounted on. For example, the Prototyping Board has RS-232 and RS-485 transceiver chips.

4.2.1 Serial Ports

There are six serial ports designated as Serial Ports A, B, C, D, E, and F. All six serial ports can operate in an asynchronous mode up to the baud rate of the system clock divided by 8. An asynchronous port can handle 7 or 8 data bits. A 9th bit address scheme, where an additional bit is sent to mark the first byte of a message, is also supported.

Serial Port A is normally used as a programming port, but may be used either as an asynchronous or as a clocked serial port once the RCM3305/RCM3315 has been programmed and is operating in the Run Mode.

Serial Port B is used to communicate with the serial flash on the RCM3305/RCM3315 and is not available for other use.

Serial Ports C and D can also be operated in the clocked serial mode. In this mode, a clock line synchronously clocks the data in or out. Either of the two communicating devices can supply the clock.

Serial Ports E and F can also be configured as HDLC serial ports. The IrDA protocol is also supported in SDLC format by these two ports.

4.3 Programming Cable

The programming cable is used to connect the programming port of the RCM3305/RCM3315 to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serail port to the CMOS voltage levels used by the Rabbit 3000.

When the **PROG** connector on the programming cable is connected to the RCM3305/RCM3315 programming port, programs can be downloaded and debugged over the serial interface.

The **DIAG** connector of the programming cable may be used on header J1 of the RCM3305/RCM3315 with the RCM3305/RCM3315 operating in the Run Mode. This allows the programming port to be used as a regular serial port.

4.3.1 Changing Between Program Mode and Run Mode

The RCM3305/RCM3315 is automatically in Program Mode when the **PROG** connector on the programming cable is attached, and is automatically in Run Mode when no programming cable is attached. When the Rabbit 3000 is reset, the operating mode is determined by the state of the SMODE pins. When the programming cable's **PROG** connector is attached, the SMODE pins are pulled high, placing the Rabbit 3000 in the Program Mode. When the programming cable's **PROG** connector is not attached, the SMODE pins are pulled low, causing the Rabbit 3000 to operate in the Run Mode.

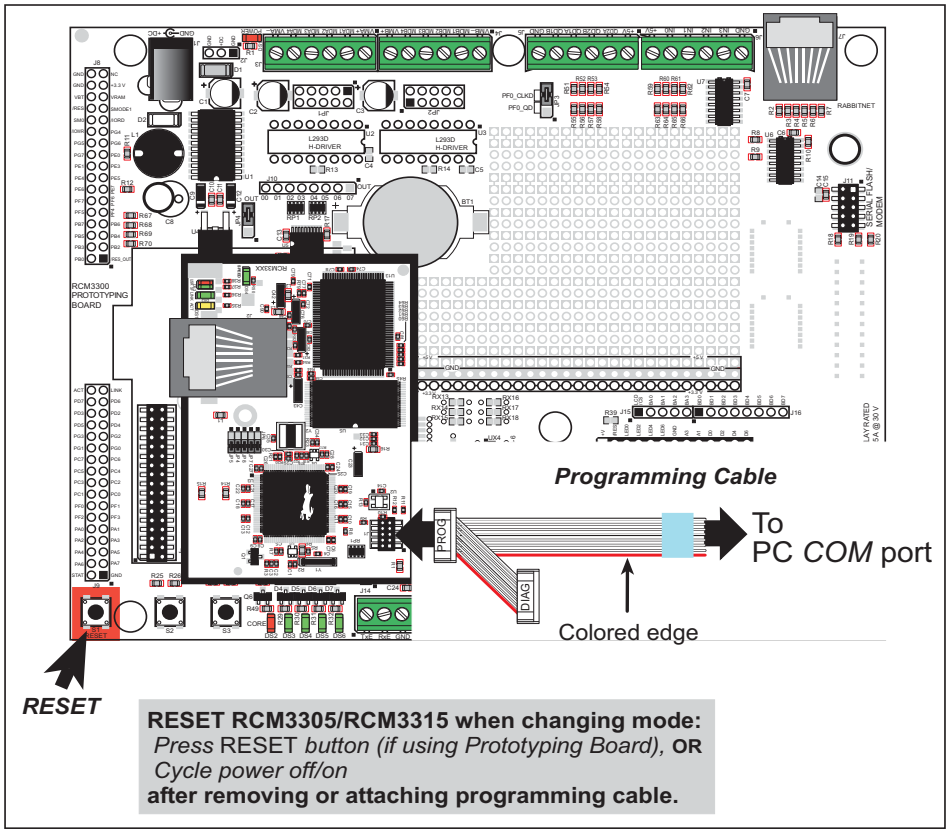


Figure 8. Switching Between Program Mode and Run Mode

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 controllers based on the Rabbit microprocessor. Chapter 5 describes the libraries and function calls related to the RCM3305/RCM3315.

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 RCM3305/RCM3315. 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: An application should be run from the program execution SRAM after the programming cable is disconnected. Your final code must always be stored in flash memory for reliable operation. RCM3305/RCM3315 modules running at 44.2 MHz have a fast program execution SRAM that is not battery-backed. Select **Code and BIOS in Flash, Run in RAM** from the Dynamic C **Options > Project Options > Compiler** menu to store the code in flash and copy it to the fast program execution SRAM at run-time to take advantage of the faster clock speed. This option optimizes the performance of RCM3305/RCM3315 modules running at 44.2 MHz.

NOTE: Do not depend on the flash memory sector size or type in your program logic. The RCM3305/RCM3315 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 2000 and later—see Rabbit's Technical Note TN257, *Running Dynamic C® With Windows Vista®*,

for additional information if you are using a Dynamic C release prior to v. 9.60 under Windows Vista. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - Exceptionally fast support for floating-point arithmetic and transcendental functions.
  - RS-232 and RS-485 serial communication.
  - Analog and digital I/O drivers.
  - I^2C, SPI, GPS, file system.
  - LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking
- Loader utility program to load binary images into Rabbit targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.
- Standard debugging features:
  - Breakpoints—Set breakpoints that can disable interrupts.
  - Single-stepping—Step into or over functions at a source or machine code level, µC/OS-II aware.
  - Code disassembly—The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
  - Watch expressions—Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
  - Register window—All processor registers and flags are displayed. The contents of general registers may be modified in the window by the user.
  - Stack window—shows the contents of the top of the stack.
  - Hex memory dump—displays the contents of memory at any address.
  - **STDIO** window—`printf` outputs to this window and keyboard input on the host PC can be detected for debugging purposes. `printf` output may also be sent to a serial port or file.

5.2.6 Prototyping Board Functions

The functions described in this section are for use with the Prototyping Board features. The source code is in the Dynamic C **SAMPLES\RCM3300\RCM33xx.LIB** library if you need to modify it for your own board design.

The **RCM33xx.LIB** library is supported by the **RN_CFG_RCM33.LIB**—library, which is used to configure the RCM3305/RCM3315 for use with RabbitNet peripheral boards on the Prototyping Board.

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

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

Call this function at the beginning of your program. This function initializes Parallel Ports A through G for use with the Prototyping Board.

Summary of Initialization

1. I/O port pins are configured for Prototyping Board operation.
2. Unused configurable I/O are set as tied inputs or outputs.
3. External I/O are disabled.
4. The LCD/keypad module is disabled.
5. RS-485 is not enabled.
6. RS-232 is not enabled.
7. LEDs are off.
8. Ethernet select is disabled.
9. Mass-storage flash select is disabled.
10. Motor control is disabled.
11. The RabbitNet SPI interface is disabled.
12. The relay is set to normally closed positions.

RETURN VALUE

None.

5.2.6.2 Digital I/O

int digIn(int channel);

Reads the input state of inputs on Prototyping Board headers J5 and J6. Do not use this function if you configure these pins for alternate use after **brdInit()** is called.

PARAMETERS

channels is the channel number corresponding to the signal on header J5 or J6

0—IN0
1—IN1
2—IN2
3—IN3
4—QD1B
5—QD1A
6—QD2B
7—QD2A

RETURN VALUE

The logic state (0 or 1) of the input.

SEE ALSO

brdInit

void digOut(int channel, int value);

Writes a value to an output channel on Prototyping Board header J10. Do not use this function if you have installed the stepper motor chips at U2 and U3.

PARAMETERS

channel is output channel 0–7 (OUT00–OUT07).

value is the value (0 or 1) to output.

RETURN VALUE

None.

SEE ALSO

brdInit

First, you need to format and partition the serial flash. Find the **FMT_DEVICE.C** sample program in the Dynamic C **SAMPLES\FileSystem** folder. Open this sample program with the **File > Open** menu, then compile and run it by pressing **F9**. **FMT_DEVICE.C** formats the mass storage device for use with the FAT file system. If the serial flash or NAND flash is already formatted, **FMT_DEVICE.C** gives you the option of erasing the mass storage flash and reformatting it with a single large partition. This erasure does not check for non-FAT partitions and will destroy *all* existing partitions.

Next, run the **INTEGRATION_FAT_SETUP.C** sample program in the Dynamic C **SAMPLES\RCM3300\Module_Integration** folder. Open this sample program with the **File > Open** menu, then compile and run it by pressing **F9**. **INTEGRATION_FAT_SETUP.C** will copy some **#ximported** files into the FAT file system.

The last step to complete before you can run the **INTEGRATION.C** sample program is to create an SSL certificate. The SSL walkthrough in the online documentation for the Dynamic C SSL module explains how to do this.

Now you are ready to run the **INTEGRATION.C** sample program in the Dynamic C **SAMPLES\RCM3300\Module_Integration** folder. Open this sample program with the **File > Open** menu, then compile and run it by pressing **F9**.

> **NOTE:** Since HTTP upload and the Dynamic C SSL module currently do not work together, compiling the **INTEGRATION.C** sample program will generate a serious warning. Ignore the warning because we are not using HTTP upload over SSL. A macro (**HTTP_UPLOAD_SSL_SUPRESS_WARNING**) is available to suppress the warning message.

Open a Web browser, and browse to the device using the IP address from the **TCP_CONFIG.LIB** library or the URL you assigned to the device. The humidity monitor will be displayed in your Web browser. This page is accessible via plain HTTP or over SSL-secured HTTPS. Click on the administrator link to bring up the admin page, which is secured automatically using SSL with a user name and a password. Use **myadmin** for user name and use **myadmin** for the password.

The admin page demonstrates some RabbitWeb capabilities and provides access to the HTTP upload page. Click the upload link to bring up the HTTP upload page, which allows you to choose new files for both the humidity monitor and the admin page. If your browser prompts you again for your user name and password, they are the same as above.

Note that the upload page is a static page included in the program flash, and can only be updated by recompiling and downloading the application. This page is protected so that you cannot accidentally change the upload page, possibly restricting yourself from performing future updates.

To try out the update capability, click the upload link on the admin page and choose a simple text file to replace **monitor.ztm**. Open another browser window and load the main page. You will see that your text file has replaced the humidity monitor. To restore the monitor, go back to the other window, click back to go to the upload page again, and choose **HUMIDITY_MONITOR.ZHTML** to replace **monitor.ztm** and click **Upload**.

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM3305/RCM3315.

Table A-1. RCM3305/RCM3315 Specifications

Parameter	RCM3305	RCM3315
Microprocessor	Low-EMI Rabbit 3000® at 44.2 MHz	
EMI Reduction	Spectrum spreader for reduced EMI (radiated emissions)	
Ethernet Port	10/100Base-T, RJ-45, 3 LEDs	
SRAM	512K program (fast SRAM) + 512K data	
Flash Memory (program)	512K	
Flash Memory (mass data storage)	8 Mbytes (serial flash)	4 Mbytes (serial flash)
LED Indicators	ACT (activity) LINK (link) SPEED (on for 100Base-T Ethernet connection) SF (serial flash) USR (user-programmable)	
Backup Battery	Connection for user-supplied backup battery (to support RTC and data SRAM)	
General-Purpose I/O	49 parallel digital I/0 lines: • 43 configurable I/O • 3 fixed inputs • 3 fixed outputs	
Additional Inputs	Startup mode (2), reset in	
Additional Outputs	Status, reset out	
External I/O Bus	Can be configured for 8 data lines and 5 address lines (shared with parallel I/O lines), plus I/O read/write	
Serial Ports	Five 3.3 V, CMOS-compatible ports (shared with I/O) • all 5 configurable as asynchronous (with IrDA) • 3 configurable as clocked serial (SPI) • 2 configurable as SDLC/HDLC • 1 asynchronous serial port dedicated for programming	
Serial Rate	Maximum asynchronous baud rate = CLK/8	
Slave Interface	A slave port allows the RCM3305/RCM3315 to be used as an intelligent peripheral device slaved to a master processor, which may either be another Rabbit 3000 or any other type of processor	
Real-Time Clock	Yes	
Timers	Ten 8-bit timers (6 cascadable, 3 reserved for internal peripherals), one 10-bit timer with 2 match registers	

A.2 Bus Loading

You must pay careful attention to bus loading when designing an interface to the RCM3305/RCM3315. This section provides bus loading information for external devices.

Table A-2 lists the capacitance for the various RCM3305/RCM3315 I/O ports.

Table A-2. Capacitance of Rabbit 3000 I/O Ports

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

Table A-3 lists the external capacitive bus loading for the various RCM3305/RCM3315 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-3.

Table A-3. External Capacitive Bus Loading -40°C to +85°C

Output Port	Clock Speed (MHz)	Maximum External Capacitive Loading (pF)
All I/O lines with clock doubler enabled	44.2	100

APPENDIX B. PROTOTYPING BOARD

Appendix B describes the features and accessories of the Prototyping Board.

B.3 Power Supply

The RCM3305/RCM3315 requires a regulated 3.15 V to 3.45 V DC power source to operate. Depending on the amount of current required by the application, different regulators can be used to supply this voltage.

The Prototyping Board has an onboard +5 V switching power regulator from which a +3.3 V linear regulator draws its supply. Thus both +5 V and +3.3 V are available on the Prototyping Board.

The Prototyping Board itself is protected against reverse polarity by a diode at D1 as shown in Figure B-3.

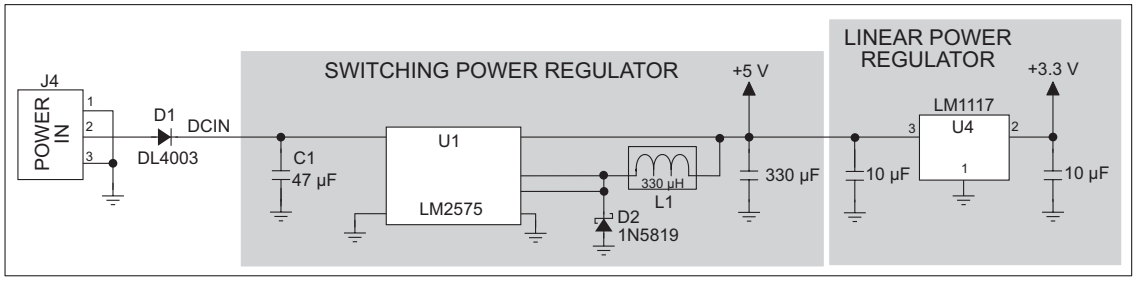


Figure B-3. Prototyping Board Power Supply

Figure B-11 shows the stepper-motor driver circuit.

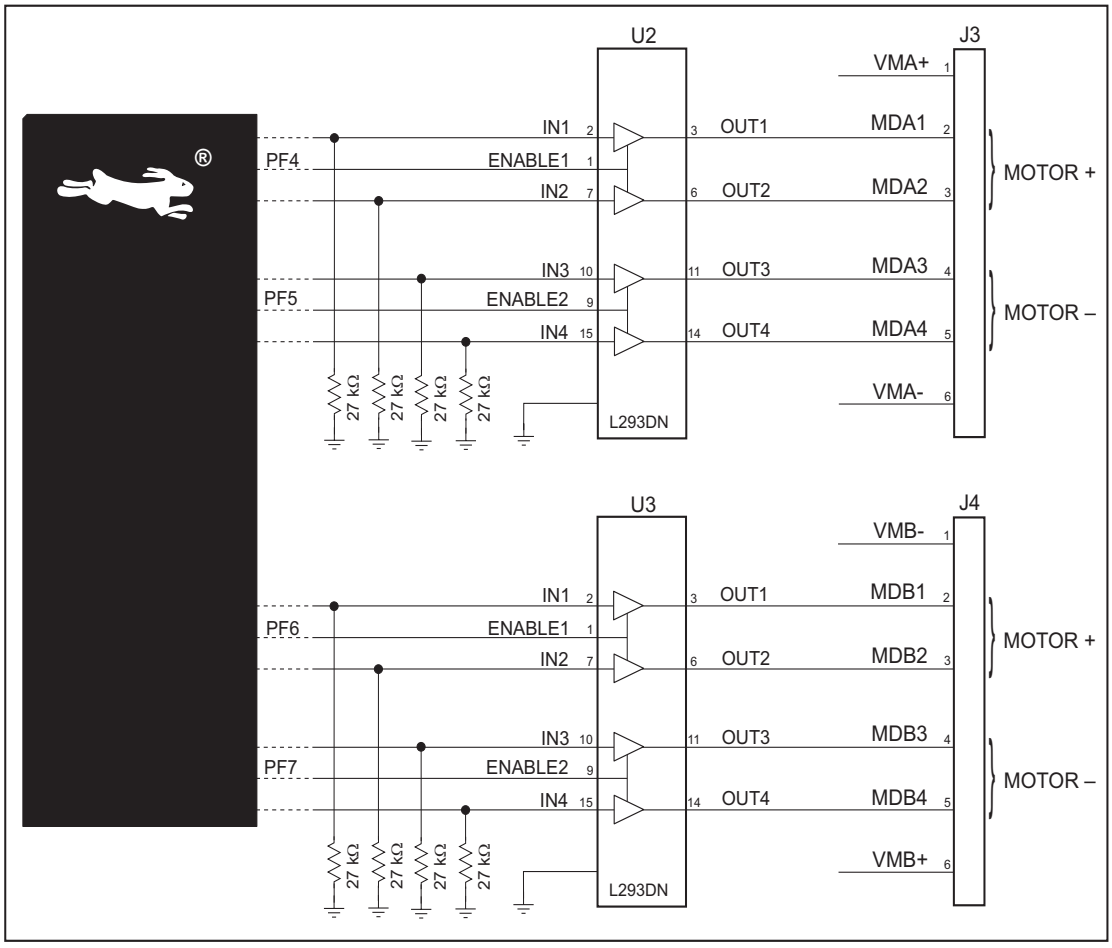


Figure B-11. Stepper-Motor Driver Circuit

The stepper motor(s) can be powered either from the onboard power supply or from an external power based on the jumper settings on headers JP1 and JP2.

Table B-3. Stepper Motor Power-Supply Options

Header	Pins Connected		Factory Default
JP1	1–2 9–10	Onboard power supply to U2	✕
	3–4 7–8	External power supply to U2	
JP2	1–2 9–10	Onboard power supply to U3	✕
	3–4 7–8	External power supply to U3	

Table B-5. Prototyping Board Use of Rabbit 3000 Parallel Ports (continued)

Port	I/O	Use		Initial State
PF0	Input	SPI, serial flash, quadrature decoder, J7		High
PF1–PF3	Input	Quadrature decoder, J7		High
PF4–PF7	Output	Motor 1–4 control		Low (disabled)
PG0	Input	Switch S1		High
PG1	Input	Switch S2		High
PG2	Input	TXF RS-232	Serial Port F	High (disabled)
PG3	Input	RXF RS-232		High (disabled)
PG4	Output	Motor driver A enable		High (disabled)
PG5	Output	Motor driver B enable		High (disabled)
PG6	Input	TXE RS-232	Serial Port E	High (disabled)
PG7	Input	RXE RS-232		High (disabled)

* Serial Port B is not available on the Prototyping Board when the RCM3305/RCM3315 is plugged in.

† PD0, PD1, and PE2 are not normally available on the Prototyping Board because they are not brought out on RCM3305 headers J3 and J4.

Mounting hardware and a 60 cm (24") extension cable are also available for the LCD/keypad module through your sales representative or authorized distributor.

Table C-1 lists the electrical, mechanical, and environmental specifications for the LCD/keypad module.

Table C-1. LCD/Keypad Specifications

Parameter	Specification
Board Size	2.60" x 3.00" x 0.75" (66 mm x 76 mm x 19 mm)
Bezel Size	4.50" × 3.60" × 0.30" (114 mm × 91 mm × 7.6 mm)
Temperature	Operating Range: 0°C to +50°C Storage Range: –40°C to +85°C
Humidity	5% to 95%, noncondensing
Power Consumption	1.5 W maximum*
Connections	Connects to high-rise header sockets on the Prototyping Board
LCD Panel Size	122 × 32 graphic display
Keypad	7-key keypad
LEDs	Seven user-programmable LEDs

* The backlight adds approximately 650 mW to the power consumption.

The LCD/keypad module has 0.1" IDC headers at J1, J2, and J3 for physical connection to other boards or ribbon cables. Figure C-2 shows the LCD/keypad module footprint. These values are relative to one of the mounting holes.

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

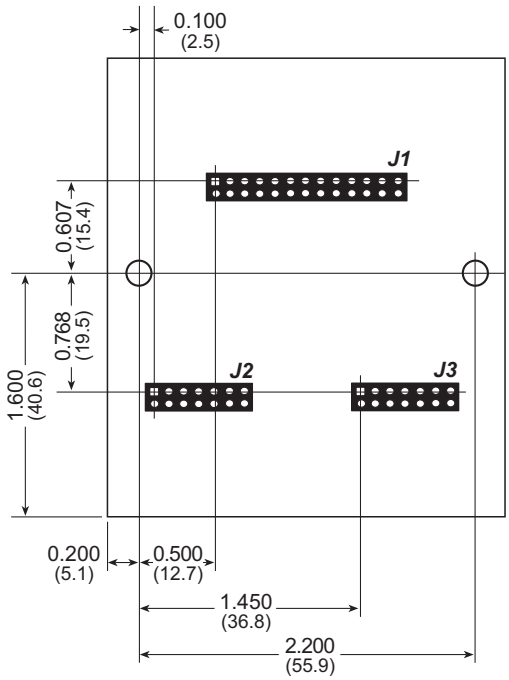


Figure C-2. User Board Footprint for LCD/Keypad Module

3. Fasten the unit with the four 4-40 screws and washers included with the LCD/keypad module. If your panel is thick, use a 4-40 screw that is approximately 3/16" (5 mm) longer than the thickness of the panel.

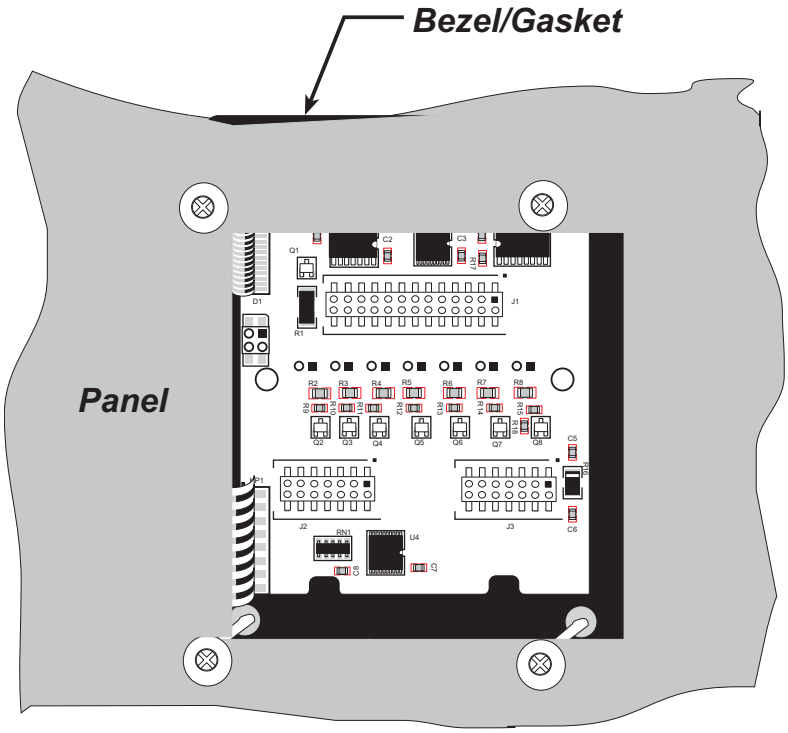


Figure C-9. LCD/Keypad Module Mounted in Panel (rear view)

Carefully tighten the screws until the gasket is compressed and the plastic bezel face-plate is touching the panel.

Do not tighten each screw fully before moving on to the next screw. Apply only one or two turns to each screw in sequence until all are tightened manually as far as they can be so that the gasket is compressed and the plastic bezel faceplate is touching the panel.

C.8 LCD/Keypad Module Function Calls

When mounted on the Prototyping Board, the LCD/keypad module uses the external I/O bus on the Rabbit 3000 chip. Remember to add the line

```
#define PORTA_AUX_IO
```

to the beginning of any programs using the external I/O bus.

C.8.1 LCD/Keypad Module Initialization

The function used to initialize the LCD/keypad module can be found in the Dynamic C **LIB\DISPLAYS\LCD122KEY7.LIB** library.

```
void dispInit();
```

Initializes the LCD/keypad module. The keypad is set up using **keypadDef()** or **keyConfig()** after this function call.

RETURN VALUE

None.

C.8.2 LEDs

When power is applied to the LCD/keypad module for the first time, the red LED (DS1) will come on, indicating that power is being applied to the LCD/keypad module. The red LED is turned off when the **brdInit** function executes.

One function is available to control the LEDs, and can be found in the Dynamic C **LIB\DISPLAYS\LCD122KEY7.LIB** library.

```
void displedOut(int led, int value);
```

LED on/off control. This function will only work when the LCD/keypad module is installed on the RCM3700 Prototyping Board.

PARAMETERS

led is the LED to control.

0 = LED DS1
1 = LED DS2
2 = LED DS3
3 = LED DS4
4 = LED DS5
5 = LED DS6
6 = LED DS7

value is the value used to control whether the LED is on or off (0 or 1).

0 = off
1 = on

RETURN VALUE

None.

void glPlotPolygon(int n, int y1, int x1, int y2, int x2, ...);

Plots the outline of a polygon in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. If fewer than 3 vertices are specified, the function will return without doing anything.

PARAMETERS

n is the number of vertices.

y1 is the *y* coordinate of the first vertex.

x1 is the *x* coordinate of the first vertex.

y2 is the *y* coordinate of the second vertex.

x2 is the *x* coordinate of the second vertex.

... are the coordinates of additional vertices.

RETURN VALUE

None.

SEE ALSO

glPlotVPolygon, glFillPolygon, glFillVPolygon

void glFillVPolygon(int n, int *pFirstCoord);

Fills a polygon in the LCD page buffer and on the LCD screen if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. If fewer than 3 vertices are specified, the function will return without doing anything.

PARAMETERS

n is the number of vertices.

pFirstCoord is a pointer to array of vertex coordinates: **x1,y1, x2,y2, x3,y3, ...**

RETURN VALUE

None.

SEE ALSO

glFillPolygon, glPlotPolygon, glPlotVPolygon