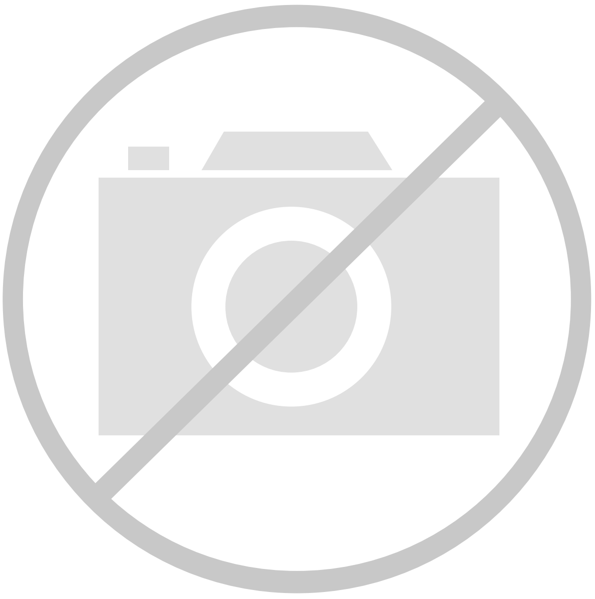
E·XFL

Digi - 20-101-1194 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 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 ~ 85°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/20-101-1194

2. Getting Started

This chapter explains how to set up and use the RCM3309/RCM3319 modules with the accompanying Prototyping Board.

NOTE: It is assumed that you have a Development Kit. If you purchased an RCM3309 or RCM3319 module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

2.1 Install Dynamic C

To develop and debug programs for the RCM3309/RCM3319 (and for all other Rabbit 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.

Dynamic C uses a COM (serial) port to communicate with the target development system. The installation allows you to choose the COM port that will be used. The default selection is COM1. Select any available USB port for Dynamic C's use. This selection can be changed later within Dynamic C.

NOTE: The installation utility does not check the selected COM port in any way. Specifying a port in use by another device (mouse, modem, etc.) may lead to a message such as **`"could not open serial port"`** when Dynamic C is started.

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

If you have purchased the optional Dynamic C Rabbit Embedded Security Pack, install it after installing Dynamic C. You must install the Rabbit Embedded Security Pack in the same directory where Dynamic C was installed.

2.2.2 Step 2 — Connect Programming Cable

The programming cable connects the RCM3309/RCM3319 to the PC running Dynamic C to download programs and to monitor the RCM3309/RCM3319 module during debugging.

Connect the 10-pin connector of the programming cable labeled **PROG** to header J1 on the RCM3309/RCM3319 as shown in Figure 3. 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.)

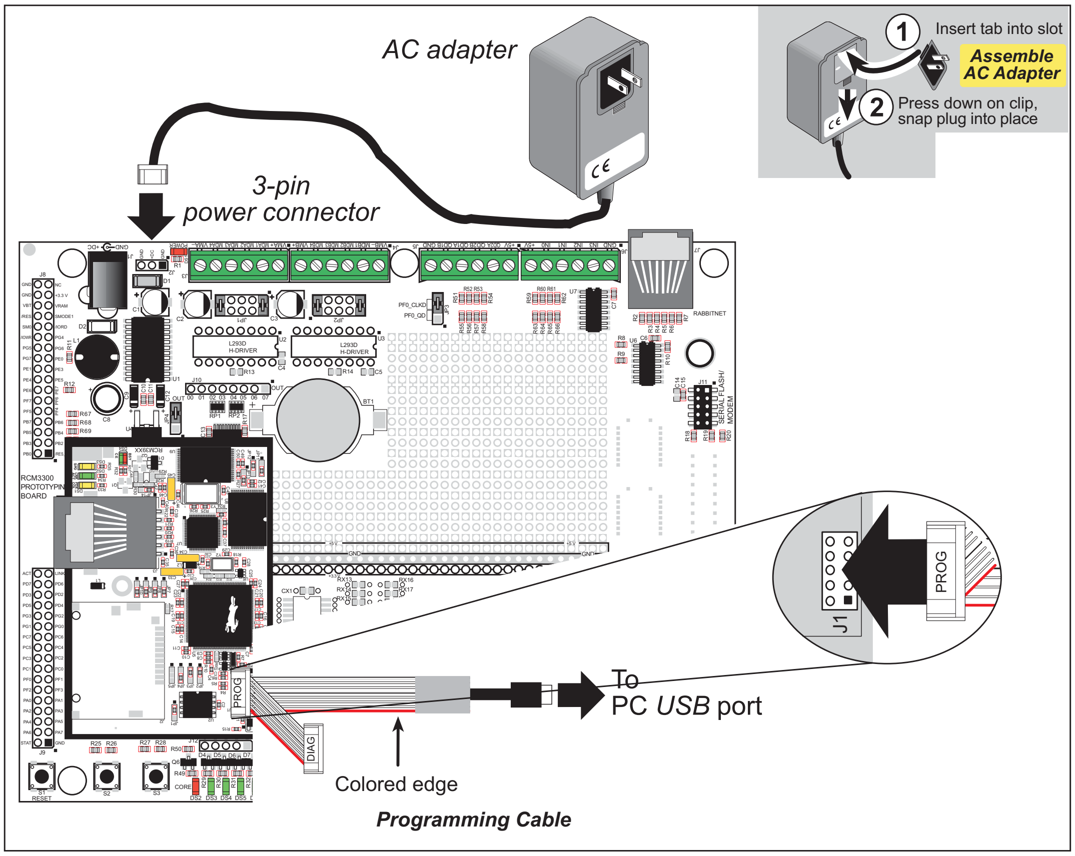


Figure 3. Connect Programming Cable and Power Supply

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.

2.2.3 Step 3 — Connect Power

When all other connections have been made, you can connect power to the Prototyping Board.

First, prepare the AC adapter for the country where it will be used by selecting the plug. The RCM3909/RCM3319 Development Kit presently includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs. Snap in the top of the plug assembly into the slot at the top of the AC adapter as shown in Figure 3, then press down on the spring-loaded clip below the plug assembly to allow the plug assembly to click into place.

Connect the AC adapter to 3-pin header J2 on the Prototyping Board as shown in Figure 3.

Plug in the AC adapter. The red **CORE** LED on the Prototyping Board should light up. The RCM3309/RCM3319 and the Prototyping Board are now ready to be used.

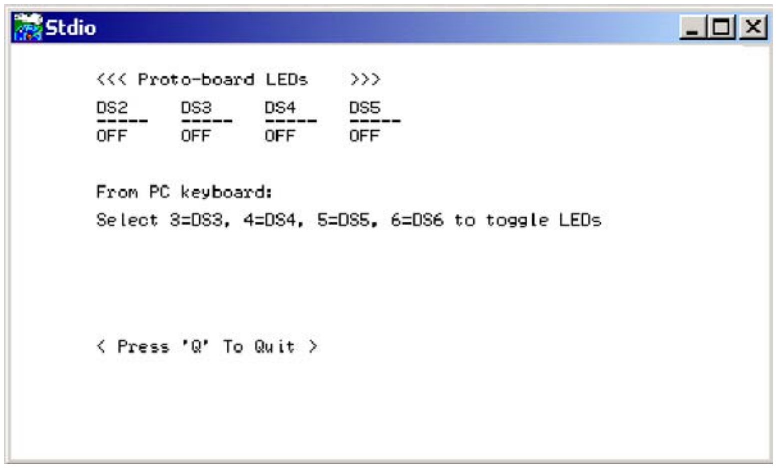
> **NOTE:** A **RESET** button is provided on the Prototyping Board to allow a hardware reset without disconnecting power.

3.2 Sample Programs

Of the many sample programs included with Dynamic C, several are specific to the RCM3309 and the RCM3319. Sample programs illustrating the general operation of the RCM3309/RCM3319, serial communication, and the serial flash are provided in the **`SAMPLES\RCM3300`** 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. Note that the RCM3309/RCM3319 must be installed on the Prototyping Board when using the sample programs described in this chapter.

- **`CONTROLLED.c`**—Demonstrates use of the digital outputs by having you turn the LEDs on the Prototyping Board on or off from the **STDIO** window on your PC.

  Once you compile and run **`CONTROLLED.C`**, the following display will appear in the Dynamic C **STDIO** window.



```
<<< Proto-board LEDs    >>>
DS2     DS3     DS4     DS5
-----   -----   -----   -----
OFF     OFF     OFF     OFF

From PC keyboard:
Select 3=DS3, 4=DS4, 5=DS5, 6=DS6 to toggle LEDs



< Press 'Q' To Quit >
```

  Press “2” or “3” or “4”or “5”on your keyboard to select LED DS3 or DS4 or DS5 or DS6 on the Prototyping Board. Then follow the prompt in the Dynamic C **STDIO** window to turn the LED on or off.

- **`FLASHLED.c`**—Demonstrates assembly-language program by flashing the USR LED on the RCM3309/RCM3319 and LEDs DS3, DS4, DS5, and DS6 on the Prototyping Board.
- **`SWRELAY.c`**—Demonstrates the relay-switching function call using the relay installed on the Prototyping Board through screw-terminal header J17.
- **`TOGGLESWITCH.c`**—Uses costatements (cooperative multitasking) to detect switches S2 and S3 using debouncing. The corresponding LEDs (DS3 and DS4) will turn on or off.

Once you have loaded and executed these five programs and have an understanding of how Dynamic C and the RCM3309/RCM3319 modules interact, you can move on and try the other sample programs, or begin building your own.

Table 2. RCM3309/RCM3319 Pinout Configurations (continued)

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J62	20	PG7	Input/Output	RXE	Serial Port E
	21	PG6	Input/Output	TXE	
	22	PG5	Input/Output	RCLKE	Serial Clock E input
	23	PG4	Input/Output	TCLKE	Serial Clock E ouput
	24	/IOWR	Output		External write strobe
	25	/IORD	Output		External read strobe
	26–27	SMODE0, SMODE1	(0,0)—start executing at address zero (0,1)—cold boot from slave port (1,0)—cold boot from clocked Serial Port A SMODE0 =1, SMODE1 = 1 Cold boot from asynchronous Serial Port A at 2400 bps (programming cable connected)		Also connected to programming cable
	28	/RESET_IN	Input		Input to Reset Generator
	29	VRAM	Output		See **Notes** below table
	30	VBAT_EXT	3 V battery Input		Minimum battery voltage 2.85 V
	31	+3.3 VIN	Power Input		3.15–3.45 V DC
	32	GND			
	33	n.c.			Reserved for future use
	34	GND			

Notes

1. When using pins 33–34 on header J3 to drive LEDs, these pins can handle a sinking current of up to 8 mA.
2. The VRAM voltage is temperature-dependent. If the VRAM voltage drops below about 1.2 V to 1.5 V, the contents of the battery-backed SRAM may be lost. If VRAM drops below 1.0 V, the 32 kHz oscillator could stop running. Pay careful attention to this voltage if you draw any current from this pin.

4.1.1 Memory I/O Interface

The Rabbit 3000 address lines (A0–A18) and all the data lines (D0–D7) are routed internally to the onboard flash memory and SRAM chips. I/O write (/IOWR) and I/O read (/IORD) are available for interfacing to external devices.

Parallel Port A can also be used as an external I/O data bus to isolate external I/O from the main data bus. Parallel Port B pins PB2–PB5 and PB7 can also be used as an auxiliary address bus.

When using the auxiliary I/O bus for a digital output or the LCD/keypad module on the Prototyping Board, or for any other reason, you must add the following line at the beginning of your program.

```
#define PORTA_AUX_IO    // required to enable auxiliary I/O bus
```

4.1.2 LEDs

The RCM3309/RCM3319 has three Ethernet status LEDs located beside the RJ-45 Ethernet jack—these are discussed in Section 4.2.

Addiitionally, there is one dual LED DS4. PD1 on the Rabbit 3000's Parallel Port D is used to enable the serial flash, and is connected to the green **CE** LED at DS4, which blinks when data are being written to or read from the serial flash. The red **BSY** LED at DS4 is a user-programmable LED, and is controlled by PD0. The **CONTROLLED.C** and **FLASHLED.C** sample programs in the Dynamic C **SAMPLES\RCM3300** folder show how to set up and use this user-programmable LED.

4.1.3 Other Inputs and Outputs

The status, /RESET_IN, SMODE0, and SMODE1 I/O are normally associated with the programming port. Since the status pin is not used by the system once a program has been downloaded and is running, the status pin can then be used as a general-purpose CMOS output. The programming port is described in more detail in Section 4.2.3.

/RES is an output from the reset circuitry that can be used to reset external peripheral devices.

- **SMTP.C**—This program demonstrates using the SMTP library to send an e-mail when the S2 and S3 switches on the Prototyping Board are pressed. LEDs DS3 and DS4 on the Prototyping Board will light up when e-mail is being sent.

6.6.1 RabbitWeb Sample Programs

The following sample programs are in the Dynamic C **SAMPLES\RCM3300\TCPIP\RABBITWEB** folder.

- **BLINKLEDS.C**—This program demonstrates a basic example to change the rate at which the DS3 and DS4 LEDs on the Prototyping Board blink.
- **DOORMONITOR.C**—The optional LCD/keypad module (see Appendix C) must be plugged in to the Prototyping Board when using this sample program. This program demonstrates adding and monitoring passwords entered via the LCD/keypad module.
- **SPRINKLER.C**—This program demonstrates how to schedule times for the relay and digital outputs in a 24-hour period.

6.6.2 Remote Application Update

The following programs that make up the featured application for the RCM3309/RCM3319 can be found in the **SAMPLES\RCM3300\RemoteApplicationUpdate** folder.

- **DLP_STATIC.C**—This program uses the TCP/IP **LIB\TCPIP\HTTP.LIB** library, and outputs a basic static Web page.
- **DLP_WEB.C**—This program outlines a basic download program with a Web interface.

Complete information on the use of these programs is provided in the ***Remote Application Update*** instructions, which are available with the online documentation.

6.6.3 Dynamic C FAT File System, RabbitWeb, and SSL Libraries

The Dynamic C FAT File System, RabbitWeb, and Secure Sockets Layer (SSL) libraries have been integrated into a sample program for the RCM3309 and the RCM3319. The sample program requires that you have installed the optional Rabbit Embedded Security Pack.

> **TIP:** Before running any of the sample programs described in this section, you should look at and run sample programs for the TCP/IP **LIB\TCPIP\ZSERVER.LIB** library, the FAT file system, RabbitWeb, SSL, the download manager, and HTTP upload to become more familiar with their operation.

The **INTEGRATION.C** sample program in the **SAMPLES\RCM3300\Module_Integration** folder demonstrates the use of the TCP/IP **LIB\TCPIP\ZSERVER.LIB** library and FAT file system functionality with RabbitWeb dynamic HTML content, all secured using SSL. The sample program also supports dynamic updates of both the application and its resources using the Rabbit Download Manager (DLM) and HTTP upload capability, respectively—note that neither of these currently supports SSL security.

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**

Figure A-4 shows a typical timing diagram for the Rabbit 3000 microprocessor external I/O read and write cycles.

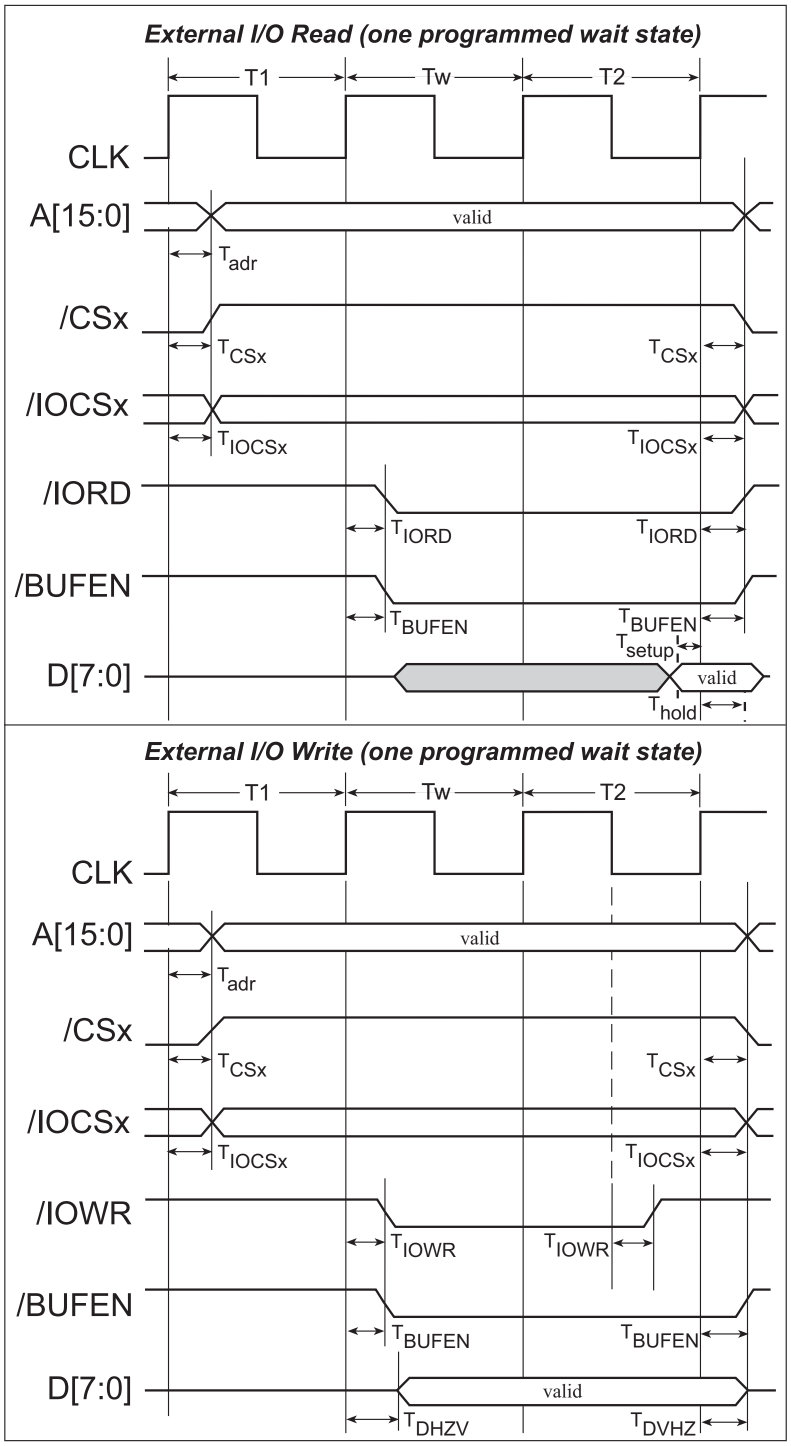


Figure A-4. I/O Read and Write Cycles—No Extra Wait States

NOTE: /IOCSx can be programmed to be active low (default) or active high.

A.5 Jumper Configurations

Figure A-5 shows the jumper locations used to configure the various RCM3309/ RCM3319 options. The black square indicates pin 1.

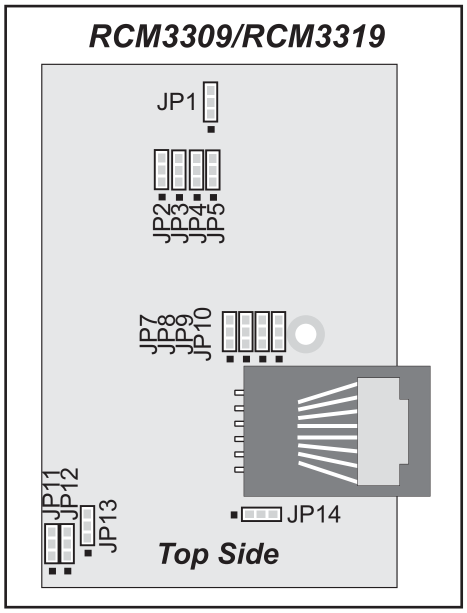


Figure A-5. Location of RCM3309/RCM3319 Configurable Positions

Table A-8 lists the configuration options.

Table A-8. RCM3309/RCM3319 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	Serial Flash Chip Enable Indicator	1–2		×
JP2	ACT or PD1 Output on J61 pin 34	1–2	ACT	×
		2–3	PD1	
JP3	LINK or PD0 Output on J61 pin 33	1–2	LINK	×
		2–3	PD0	
JP4	ENET or PE0 Output on J62 pin 19	1–2	ENET	
		2–3	PE0	×
JP5	NAND Flash Chip Enable	1–2	Reserved for future use	n.c.
		2–3	PD1 controls NAND Flash	

- **Module Extension Headers**—The complete pin set of the RCM3309/RCM3319 module is duplicated at headers J8 and J9. Developers can solder wires directly into the appropriate holes, or, for more flexible development, 2 × 17 header strips with a 0.1" pitch can be soldered into place. See Figure B-4 for the header pinouts.
- **Digital I/O**—Four digital inputs are available on screw-terminal header J6. See Figure B-4 for the header pinouts.
- **RS-232**—Two 3-wire serial ports or one 5-wire RS-232 serial port are available on the Prototyping Board at screw-terminal header J14.
- **RS-485**—One RS-485 serial port is available on the Prototyping Board at screw-terminal header J14.
- **Quadrature Decoder**—Four quadrature decoder inputs (PF0–PF3) from the Rabbit 3000 chip are available on screw-terminal header J5. See Figure B-4 for the header pinouts.
- **H-Bridge Motor Driver**—Two pairs of H-bridge motor drivers are supported using screw-terminal headers J3 and J4 on the Prototyping Board for stepper-motor control. See Figure B-4 for the header pinouts.
- **RabbitNet Port**—One RS-422 RabbitNet port (shared with the serial flash interface) is available to allow RabbitNet peripheral cards to be used with the Prototyping Board.
- **Serial Flash Interface**—One serial flash interface (shared with the RabbitNet port) is available to allow Rabbit’s SF1000 series serial flash to be used on the Prototyping Board.

The Prototyping Board comes with a 220 Ω termination resistor and two 681 Ω bias resistors installed and enabled with jumpers across pins 1–2 and 5–6 on header JP5, as shown in Figure B-9.

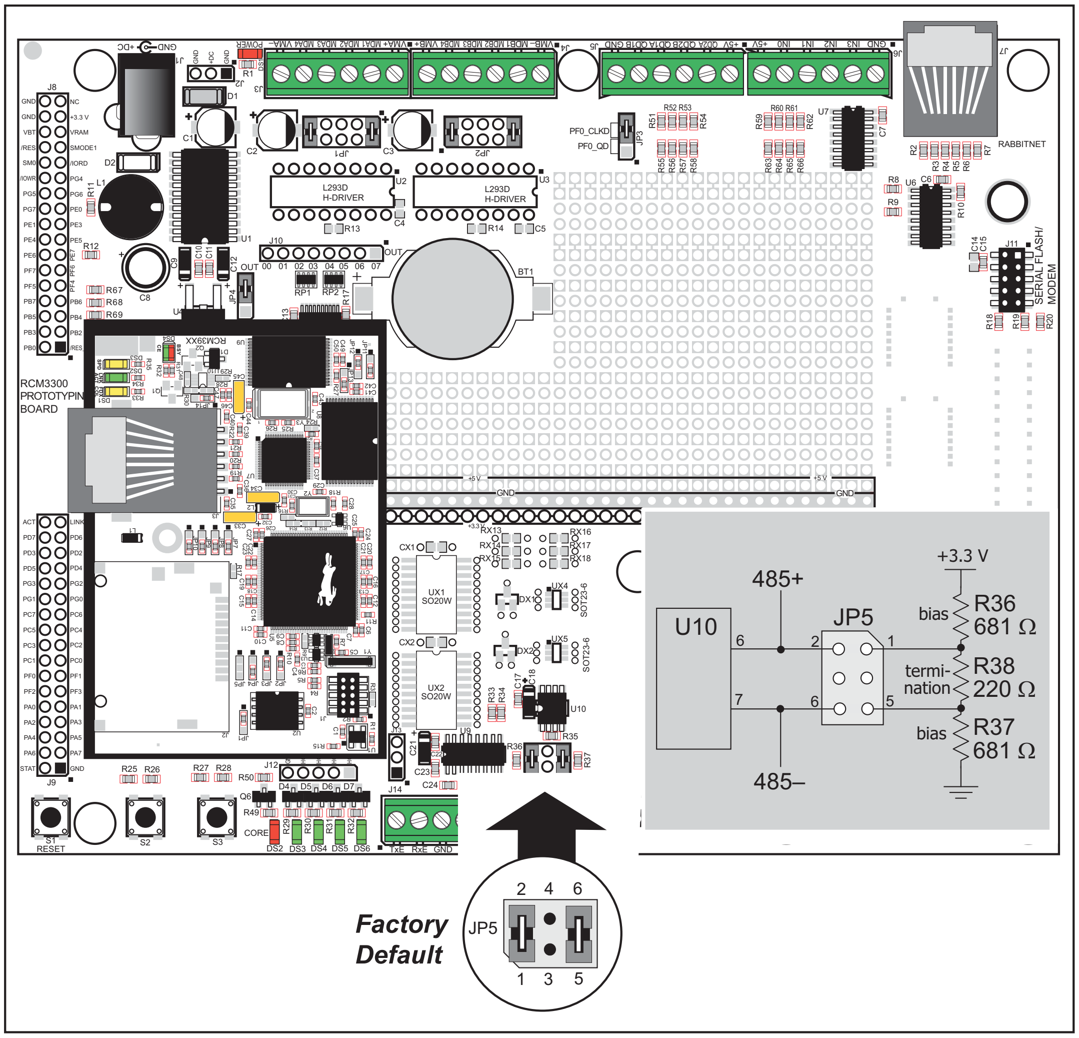


Figure B-9. RS-485 Termination and Bias Resistors

For best performance, the termination resistors in a multidrop network should be enabled only on the end nodes of the network, but ***not*** on the intervening nodes. Jumpers on boards whose termination resistors are not enabled may be stored across pins 1–3 and 4–6 of header JP5.

B.4.7 RabbitNet Ports

The RJ-45 jack labeled ***RabbitNet*** is a clocked SPI RS-422 serial I/O expansion port for use with RabbitNet peripheral boards. The ***RabbitNet*** jack does ***not*** support Ethernet connections. Header JP3 must have pins 2–3 jumpered when using the RabbitNet port.

The RabbitNet port is enabled in software by setting PD2 = 1. Note that the RabbitNet port and the J11 interface cannot be used simultaneously.

Table B-4 lists the configuration options using jumpers.

Table B-4. Prototyping Board Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	Stepper Motor Power-Supply Options (U2)	1–2 9–10	Onboard power supply	×
		3–4 7–8	External power supply	
JP2	Stepper Motor Power-Supply Options (U3)	1–2 9–10	Onboard power supply	×
		3–4 7–8	External power supply	
JP3	PF0 Option	1–2	Quadrature decoder inputs enabled	
		2–3	RabbitNet/Serial Flash interface enabled	×
JP4	RCM3309/RCM3319 Power Supply	2–3	RCM3309/RCM3319 powered via Prototyping Board	×
JP5	RS-485 Bias and Termination Resistors	1–2 5–6	Bias and termination resistors connected	×
		1–3 4–6	Bias and termination resistors ***not*** connected (parking position for jumpers)	

C.3 Keypad Labeling

The keypad may be labeled according to your needs. A template is provided in Figure C-4 to allow you to design your own keypad label insert.

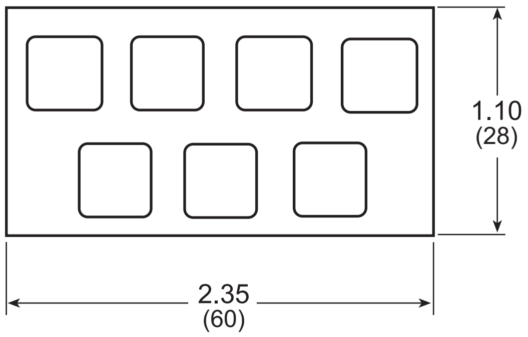


Figure C-4. Keypad Template

To replace the keypad legend, remove the old legend and insert your new legend prepared according to the template in Figure C-4. The keypad legend is located under the blue keypad matte, and is accessible from the left only as shown in Figure C-5.

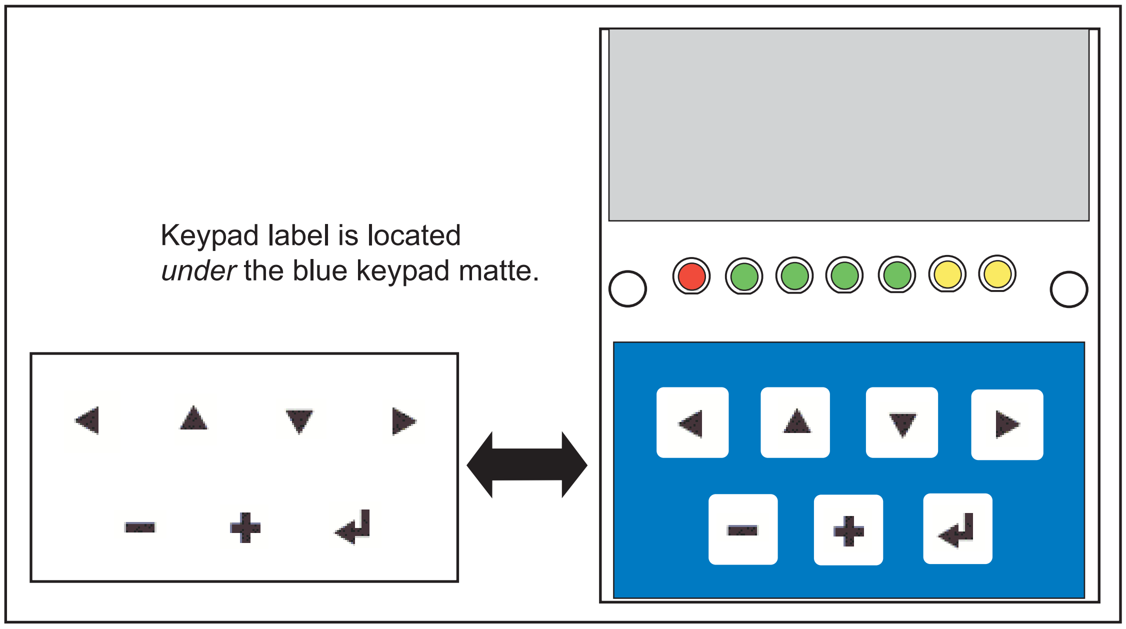


Figure C-5. Removing and Inserting Keypad Label

The sample program **`KEYBASIC.C`** in the **`122x32_1x7`** folder in **`SAMPLES\LCD_KEYPAD`** shows how to reconfigure the keypad for different applications.

C.8.3 LCD Display

The functions used to control the LCD display are contained in the **`GRAPHIC.LIB`** library located in the Dynamic C **`LIB\DISPLAYS\GRAPHIC`** library folder. When *x* and *y* coordinates on the display screen are specified, *x* can range from 0 to 121, and *y* can range from 0 to 31. These numbers represent pixels from the top left corner of the display.

glInit

```
void glInit(void);
```

DESCRIPTION

Initializes the display devices, clears the screen.

RETURN VALUE

None.

SEE ALSO

`glDispOnOFF, glBacklight, glSetContrast, glPlotDot, glBlock, glPlotDot, glPlotPolygon, glPlotCircle, glHScroll, glVScroll, glXFontInit, glPrintf, glPutChar, glSetBrushType, glBuffLock, glBuffUnlock, glPlotLine`

glBackLight

```
void glBackLight(int onOff);
```

DESCRIPTION

Turns the display backlight on or off.

PARAMETER

`onOff`	turns the backlight on or off 1—turn the backlight on 0—turn the backlight off

RETURN VALUE

None.

SEE ALSO

`glInit, glDispOnoff, glSetContrast`

glFontCharAddr

```
unsigned long glFontCharAddr(fontInfo *pInfo, char letter);
```

DESCRIPTION

Returns the **`xmem`** address of the character from the specified font set.

PARAMETERS

`pInfo`	pointer to the **`xmem`** address of the bitmap font set.
`letter`	an ASCII character.

RETURN VALUE

`xmem` address of bitmap character font, column major and byte-aligned.

SEE ALSO

`glPutFont, glPrintf`

glPutFont

```
void glPutFont(int x, int y, fontInfo *pInfo, char code);
```

DESCRIPTION

Puts an entry from the font table to the page buffer and on the LCD if the buffer is unlocked. Each font character's bitmap is column major and byte-aligned. Any portion of the bitmap character that is outside the LCD display area will be clipped.

PARAMETERS

`x`	the *x* coordinate (column) of the top left corner of the text.
`y`	the *y* coordinate (row) of the top left corner of the text.
`pInfo`	a pointer to the font descriptor.
`code`	the ASCII character to display.

RETURN VALUE

None.

SEE ALSO

`glFontCharAddr, glPrintf`

glGetBrushType

```
int glGetBrushType(void);
```

DESCRIPTION

Gets the current method (or color) of pixels drawn by subsequent graphic calls.

RETURN VALUE

The current brush type.

SEE ALSO

`glSetBrushType`

glXGetBitmap

```
void glXGetBitmap(int x, int y, int bmWidth, int bmHeight,
   unsigned long xBm);
```

DESCRIPTION

Gets a bitmap from the LCD page buffer and stores it in **`xmem`** RAM. This function automatically calls **`glXGetFastmap`** if the left edge of the bitmap is byte-aligned and the left edge and width are each evenly divisible by 8.

This function call is intended for use only when a graphic engine is used to interface with the LCD/keypad module.

PARAMETERS

`x`	the *x* coordinate in pixels of the top left corner of the bitmap (*x* must be evenly divisible by 8).
`y`	the *y* coordinate in pixels of the top left corner of the bitmap.
`bmWidth`	the width in pixels of the bitmap (must be evenly divisible by 8).
`bmHeight`	the height in pixels of the bitmap.
`xBm`	the **`xmem`** RAM storage address of the bitmap.

RETURN VALUE

None.

`TextBorder`

`void TextBorder(windowFrame *wPtr);`

DESCRIPTION

This function displays the border for a given window frame. This function will automatically adjust the text window parameters to accommodate the space taken by the text border. This adjustment will only occur once after the **`TextBorderInit()`** function executes.

NOTE: Execute the **`TextWindowFrame()`** function before using this function.

PARAMETER

`wPtr`	a pointer to the window frame descriptor.

RETURN VALUE

None.

SEE ALSO

`TextBorderInit, TextGotoXY, TextPutChar, TextWindowFrame, TextCursorLocation`

`TextGotoXY`

`void TextGotoXY(windowFrame *window, int col, int row);`

DESCRIPTION

Sets the cursor location to display the next character. The display location is based on the height and width of the character to be displayed.

NOTE: Execute the **`TextWindowFrame()`** function before using this function.

PARAMETERS

`window`	a pointer to a font descriptor.
`col`	a character column location.
`row`	a character row location.

RETURN VALUE

None.

SEE ALSO

`TextPutChar, TextPrintf, TextWindowFrame`

`TextCursorLocation`

```
void TextCursorLocation(windowFrame *window, int *col, int *row);
```

DESCRIPTION

Gets the current cursor location that was set by a graphic **`Text...`** function.

NOTE: Execute the **`TextWindowFrame()`** function before using this function.

PARAMETERS

`window`	a pointer to a font descriptor.
`col`	a pointer to cursor column variable.
`row`	a pointer to cursor row variable.

RETURN VALUE

Lower word = Cursor Row location
Upper word = Cursor Column location

SEE ALSO

`TextGotoXY, TextPrintf, TextWindowFrame, TextCursorLocation`

rn_read

`int rn_read(int handle, int regno, char *recdata, int datalen);`

DESCRIPTION

Reads a string from the specified device and register. Waits for results. This function will check device information to determine that the peripheral card is connected to a master.

PARAMETERS

`handle`	is an address index to device information. Use **`rn_device()`** or **`rn_find()`** to establish the handle.
`regno`	is the command register number as designated by each device.
`recdata`	is a pointer to the address of the string to read from the device.
`datalen`	is the number of bytes to read (0–15).

NOTE: A data length of 0 will transmit the one-byte command register number.

RETURN VALUE

The status byte from the previous command.

-1 means that device information indicates the peripheral card is not connected to the master, and -2 means that the data length was greater than 15.

SEE ALSO

`rn_write`