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	22.1MHz
Flash Size	512KB
RAM Size	512KB
Connector Type	IDC Header 2x20
Size / Dimension	1.23" x 2.11" (31mm x 54mm)
Operating Temperature	-40°C ~ 85°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/101-0672

1.2 Advantages of the RCM3600

- Fast time to market using a fully engineered, “ready-to-run/ready-to-program” microprocessor core.
- 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.

1.3 Development and Evaluation Tools

1.3.1 Development Kit

The Development Kit contains the hardware you need to use your RCM3600 module.

- RCM3600 module.
- Prototyping Board.
- AC adapter, 12 V DC, 500 mA (included only with Development Kits sold for the North American market). A header plug leading to bare leads is provided to allow overseas users to connect their own power supply with a DC output of 7.5–30 V.
- Programming cable with 10-pin header and DB9 connections, and integrated level-matching circuitry.
- Cable kits to access RS-485 and analog input connectors on Prototyping Board.
- ***Dynamic C*** CD-ROM, with complete product documentation on disk.
- ***Getting Started*** instructions.
- Accessory parts for use on the Prototyping Board.
- ***Rabbit 3000 Processor Easy Reference*** poster.
- Registration card.

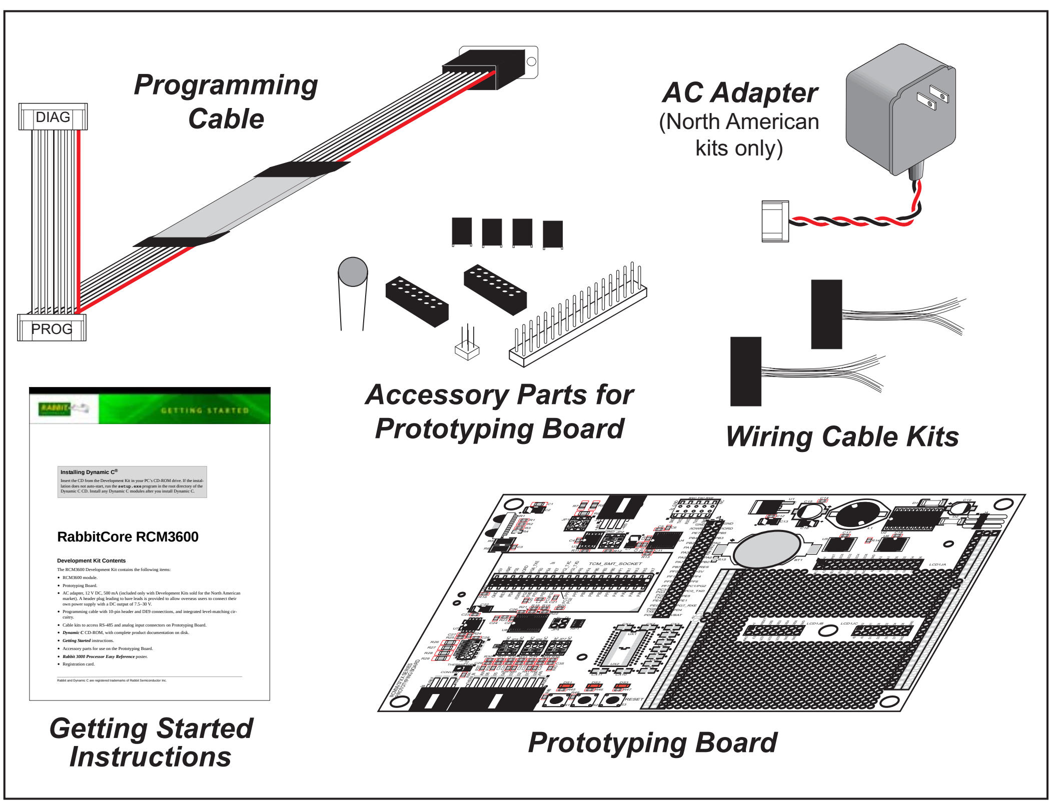


Figure 1. RCM3600 Development Kit

2.2 Hardware Connections

There are three steps to prepare the RCM3600 for use with Dynamic C and the sample programs:

1. Attach the RCM3600 module to the Prototyping Board.
2. Connect the programming cable between the RCM3600 and the COM port on the workstation PC.
3. Connect the power supply to the Prototyping Board.

2.2.1 Attach Module to Prototyping Board

Turn the RCM3600 module so that the Rabbit 3000 chip is facing up as shown in Figure 2 below. Insert the pins from the module's J1 header on the bottom side of the RCM3600 into the TCM_SMT_SOCKET socket on the Prototyping Board. The shaded corner notch at the bottom right corner of the RCM3600 module should face the same direction as the corresponding notch below it on the Prototyping Board.

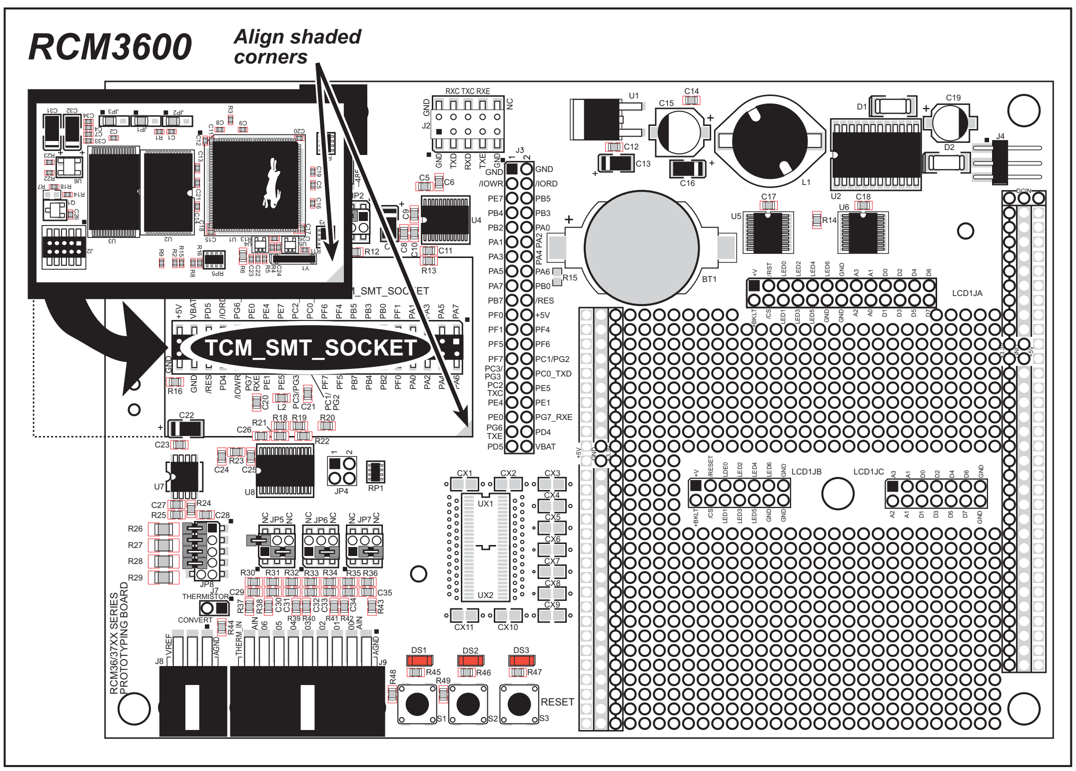


Figure 2. Install the RCM3600 Series on the Prototyping Board

NOTE: It is important that you line up the pins on header J1 of the RCM3600 module exactly with the corresponding pins of the TCM_SMT_SOCKET socket 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.

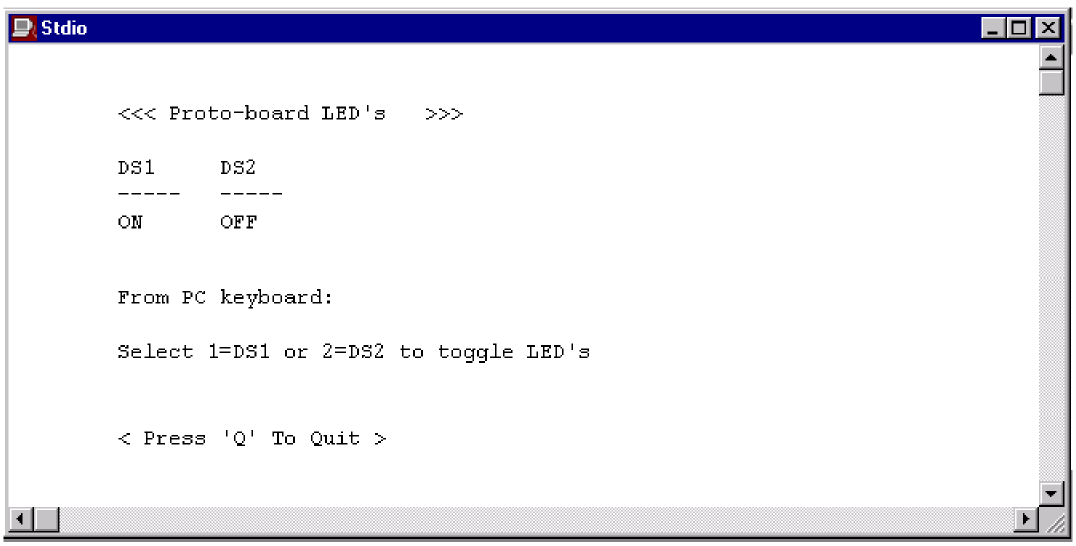
Press the module's pins firmly into the Prototyping Board headers.

3.2 Sample Programs

Of the many sample programs included with Dynamic C, several are specific to the RCM3600. Sample programs illustrating the general operation of the RCM3600, serial communication, and the A/D converter on the Prototyping Board are provided in the **SAMPLES\RCM3600** 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 RCM3600 must be installed on the Prototyping Board when using these sample programs. Sample programs for the optional LCD/keypad module are described in Appendix C.

- **CONTROLLED.c**—Demonstrates use of the digital inputs 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.



  Press “1” or “2” on your keyboard to select LED DS1 or DS2 on the Prototyping Board. Then follow the prompt in the Dynamic C **STDIO** window to turn the LED on or off.

- **FLASHLED.c**—Demonstrates the use of assembly language to flash LEDs DS1 and DS2 on the Prototyping Board at different rates. Once you have compiled and run this program, LEDs DS1 and DS2 will flash on/off at different rates.

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/0 write (/IOWR) and I/0 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 either Ethernet 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 Other Inputs and Outputs

/RES is an output from the reset circuitry that can be used to reset other peripheral devices. This pin can also be used to reset the microprocessor.

4.5 Memory

4.5.1 SRAM

RCM3600 series boards have 256K–512K of SRAM.

4.5.2 Flash EPROM

RCM3600 series boards also have 256K–512K of flash EPROM.

> **NOTE:** Rabbit Semiconductor recommends that any customer applications should not be constrained by the sector size of the flash EPROM since it may be necessary to change the sector size in the future.

Writing to arbitrary flash memory addresses at run time is also discouraged. Instead, use a portion of the “user block” area to store persistent data. The function calls `writeUserBlock` and `readUserBlock` are provided for this. Refer to the ***Rabbit 3000 Microprocessor Designer’s Handbook*** for additional information.

A Flash Memory Bank Select jumper configuration option based on 0 Ω surface-mounted resistors exists at header JP1 on the RCM3600 modules. This option, used in conjunction with some configuration macros, allows Dynamic C to compile two different co-resident programs for the upper and lower halves of the 512K flash in such a way that both programs start at logical address 0000. This is useful for applications that require a resident download manager and a separate downloaded program. See Technical Note TN218, ***Implementing a Serial Download Manager for a 256K Flash***, for details.

4.5.3 Dynamic C BIOS Source Files

The Dynamic C BIOS source files handle different standard RAM and flash EPROM sizes automatically.

`unsigned int anaIn(unsigned int channel, int opmode, int gaincode);`

Reads the value of an analog input channel using the direct method of addressing the ADS7870 A/D converter. The A/D converter is enabled the first time this function is called—this will take approximately 1 second to ensure that the A/D converter capacitor is fully charged.

PARAMETERS

`channel` is the channel number (0 to 7) corresponding to ADC_IN0 to ADC_IN7

`opmode` is the mode of operation:

`SINGLE`—single-ended input
`DIFF`—differential input
`mAMP`—4–20 mA input

`channel`	`SINGLE`	`DIFF`	`mAMP`
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*

* Not accessible on RCM3600 Prototyping Board.

`gaincode` is the gain code of 0 to 7

Gain Code	Multiplier	Voltage Range* (V)
0	x1	0–20
1	x2	0–10
2	x4	0–5
3	x5	0–4
4	x8	0–2.5
5	x10	0–2
6	x16	0–1.25
7	x20	0–1

* Applies to RCM3600 Prototyping Board.

RETURN VALUE

A value corresponding to the voltage on the analog input channel:

0–2047 for 11-bit A/D conversions (signed 12th bit)
`ADOVERFLOW` (defined macro = -4096) if overflow or out of range
-4095 if conversion is incomplete or busy-bit timeout

SEE ALSO

`anaIn, anaInConfig, anaInDriver`

`int anaInCalib(int channel, int opmode, int gaincode, int value1, float volts1, int value2, float volts2);`

Calibrates the response of the desired A/D converter channel as a linear function using the two conversion points provided. Four values are calculated and placed into global tables to be later stored into simulated EEPROM using the function **`anaInEEWr()`**. Each channel will have a linear constant and a voltage offset.

PARAMETERS

`channel` is the analog input channel number (0 to 7) corresponding to ADC_IN0 to ADC_IN7

`opmode` is the mode of operation:

`SINGLE`—single-ended input
`DIFF`—differential input
`mAMP`—milliamp input

`channel`	**`SINGLE`**	**`DIFF`**	**`mAMP`**
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*

* Not accessible on Prototyping Board.

`gaincode` is the gain code of 0 to 7

Gain Code	Multiplier	Voltage Range* (V)
0	x1	0–20
1	x2	0–10
2	x4	0–5
3	x5	0–4
4	x8	0–2.5
5	x10	0–2
6	x16	0–1.25
7	x20	0–1

* Applies to RCM3600 Prototyping Board.

B.1.1 Prototyping Board Features

- **Power Connection**—A 3-pin header is provided for connection to the power supply. Note that the 3-pin header is symmetrical, with both outer pins connected to ground and the center pin connected to the raw DCIN input. The cable of the AC adapter provided with the North American version of the Development Kit ends in a plug that connects to the power-supply header, and can be connected to the 3-pin header in either orientation. A similar header plug leading to bare leads is provided for overseas customers.

  Users providing their own power supply should ensure that it delivers 7.5–30 V DC at 500 mA. The voltage regulators will get warm while in use.

- **Regulated Power Supply**—The raw DC voltage provided at the POWER IN power-input jack is routed to a 5 V switching voltage regulator, then to a separate 3.3 V linear regulator. The regulators provide stable power to the RCM3600 module and the Prototyping Board.

- **Power LED**—The power LED lights whenever power is connected to the Prototyping Board.

- **Reset Switch**—A momentary-contact, normally open switch is connected directly to the RCM3600's **/RESET_IN** pin. Pressing the switch forces a hardware reset of the system.

- **I/O Switches and LEDs**—Two momentary-contact, normally open switches are connected to the PF4 and PB7 pins of the RCM3600 module and may be read as inputs by sample applications.

  Two LEDs are connected to the PF6 and PF7 pins of the RCM3600 module, and may be driven as output indicators by sample applications.

- **Prototyping Area**—A generous prototyping area has been provided for the installation of through-hole components. +3.3 V, +5 V, and Ground buses run at both edges of this area. Several areas for surface-mount devices are also available. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.) Each SMT pad is connected to a hole designed to accept a 30 AWG solid wire.

- **LCD/Keypad Module**—Rabbit Semiconductor's LCD/keypad module may be plugged in directly to headers LCD1JA, LCD1JB, and LCD1JC. The signals on headers LCD1JB and LCD1JC will be available only if the LCD/keypad module is plugged in to header LCD1JA. Appendix C provides complete information for mounting and using the LCD/keypad module.

- **Module Extension Headers**—The complete non-analog pin set of the RCM3600 module is duplicated at header J3. Developers can solder wires directly into the appropriate holes, or, for more flexible development, a 2 x 20 header strip with a 0.1" pitch can be soldered into place. See Figure B-4 for the header pinouts.

- **Analog I/O Shrouded Headers**—The complete analog pin set of the RCM3600 Prototyping Board is available on shrouded headers J8 and J9. See Figure B-4 for the header pinouts.

- **RS-232**—Three 3-wire serial ports or one 5-wire RS-232 serial port and one 3-wire serial port are available on the Prototyping Board at header J2. A jumper on header JP2 is used to select the drivers for Serial Port E, which can be set either as a 3-wire RS-232 serial port or as an RS-485 serial port. Serial Ports C and D are not available while the IrDA transceiver is in use.

  A 10-pin 0.1-inch spacing header strip is installed at J2 allows you to connect a ribbon cable that leads to a standard DE9 serial connector.

- **RS-485**—One RS-485 serial port is available on the Prototyping Board at shrouded header J1. A 3-pin shrouded header is installed at J1. A jumper on header JP2 enables the RS-485 output for Serial Port E.

- **IrDA**—An infrared transceiver is included on the Prototyping Board, and is capable of handling link distances up to 1.5 m. The IrDA uses Serial Port F—Serial Ports C and D are unavailable while Serial Port F is in use.

B.4.2.4 A/D Converter Calibration

To get the best results from the A/D converter, it is necessary to calibrate each mode (single-ended, differential, and current) for each of its gains. It is imperative that you calibrate each of the A/D converter inputs in the same manner as they are to be used in the application. For example, if you will be performing floating differential measurements or differential measurements using a common analog ground, then calibrate the A/D converter in the corresponding manner. The calibration must be done with the attenuator reference selection jumper in the desired position (see Figure B-5). If a calibration is performed and the jumper is subsequently moved, the corresponding input(s) must be recalibrated. The calibration table in software only holds calibration constants based on mode, channel, and gain. ***Other factors affecting the calibration must be taken into account by calibrating using the same mode and gain setup as in the intended use.***

Sample programs are provided to illustrate how to read and calibrate the various A/D inputs for the three operating modes.

Mode	Read	Calibrate
Single-Ended, one channel	—	**AD_CALSE_CH.C**
Single-Ended, all channels	**AD_RDSE_ALL.C**	**AD_CALSE_ALL.C**
Milliamp, one channel	**AD_RDMA_CH.C**	**AD_CALMA_CH.C**
Differential, analog ground	**AD_RDDIFF_CH.C**	**AD_CALDIFF_CH.C**

These sample programs are found in the Dynamic C **SAMPLES\RCM3600\ADC** subdirectory. See Section 3.2.2 for more information on these sample programs and how to use them.

B.4.3.2 RS-485

The RCM3600 Prototyping Board has one RS-485 serial channel, which is connected to the Rabbit 3000 Serial Port E through an RS-485 transceiver. The half-duplex communication uses an output from PF5 on the Rabbit 3000 to control the transmit enable on the communication line. Using this scheme a strict master/slave relationship must exist between devices to insure that no two devices attempt to drive the bus simultaneously.

Serial Port E is configured in software for RS-485 as follows.

```
#define ser485open serEopen
#define ser485close serEclose
#define ser485wrFlush serEwrFlush
#define ser485rdFlush serErdFlush
#define ser485putc serEputc
#define ser485getc serEgetc

#define EINBUFSIZE  15
#define EOUTBUFSIZE 15
```

The configuration shown above is based on circular buffers. RS-485 configuration may also be done using functions from the **PACKET.LIB** library.

The RCM3600 Prototyping Boards with RCM3600 modules installed can be used in an RS-485 multidrop network spanning up to 1200 m (4000 ft), and there can be as many as 32 attached devices. Connect the 485+ to 485+ and 485– to 485– using single twisted-pair wires as shown in Figure B-8. Note that a common ground is recommended.

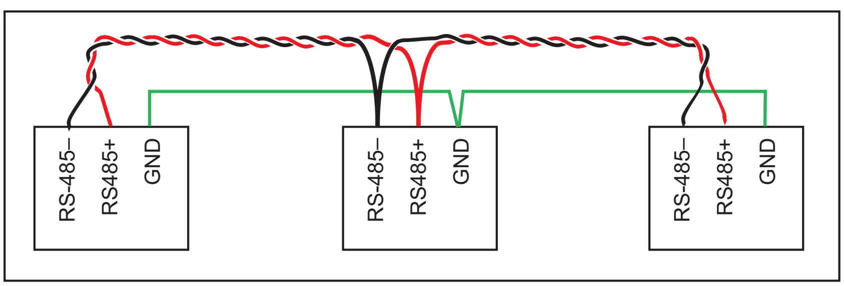


Figure B-8. RCM3600 Multidrop Network

The RCM3600 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 JP1, as shown in Figure B-9.

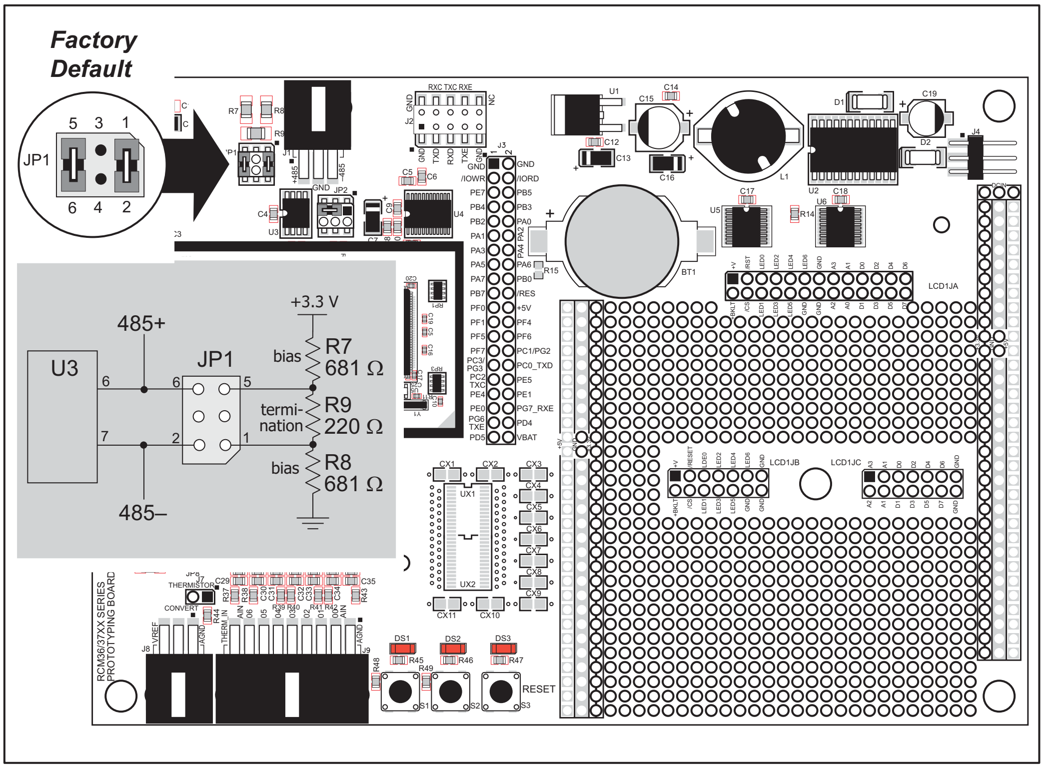


Figure B-9. RS-485 Termination and Bias Resistors

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 JP1.

B.4.4 Other Prototyping Board Modules

An optional LCD/keypad module is available that can be mounted on the Prototyping Board. The signals on headers LCD1JB and LCD1JC will be available only if the LCD/keypad module is installed. Refer to Appendix C, “LCD/Keypad Module,” for complete information.

> **CAUTION:** Pin PB7 is connected as both switch S2 and as an external I/O bus on the Prototyping Board. Do not use S2 when the LCD/keypad module is installed.

C.5 Install Connectors on Prototyping Board

Before you can use the LCD/keypad module with the RCM3600 Prototyping Board, you will need to install connectors to attach the LCD/keypad module to the RCM3600 Prototyping Board. These connectors are included with the RCM3600 Development Kit.

First solder the 2 x 13 connector to location LCD1JA on the RCM3600 Prototyping Board as shown in Figure C-7.

- If you plan to bezel-mount the LCD/keypad module, continue with the bezel-mounting instructions in Section C.7, “Bezel-Mount Installation.”
- If you plan to mount the LCD/keypad module directly on the RCM3600 Prototyping Board, solder two additional 2 x 7 connectors at locations LCD1JB and LCD1JC on the RCM3600 Prototyping Board. Section C.6, “Mounting LCD/Keypad Module on the Prototyping Board,” explains how to mount the LCD/keypad module on the RCM3600 Prototyping Board.

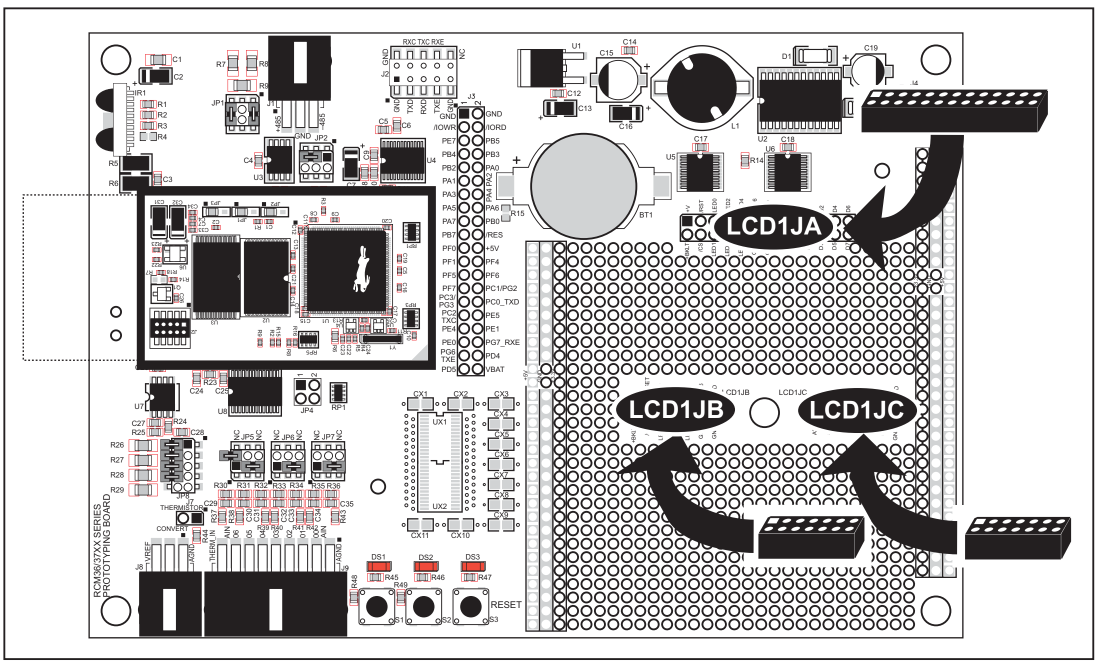


Figure C-7. Solder Connectors to RCM3600 Prototyping Board

void glPutChar(char ch, char *ptr, int *cnt, glPutCharInst *pInst)

Provides an interface between the **STDIO** string-handling functions and the graphic library. The **STDIO** string-formatting function will call this function, one character at a time, until the entire formatted string has been parsed. Any portion of the bitmap character that is outside the LCD display area will be clipped.

PARAMETERS

ch is the character to be displayed on the LCD.

***ptr** is not used, but is a place holder for **STDIO** string functions.

***cnt** is not used, is a place holder for **STDIO** string functions.

pInst is a pointer to the font descriptor.

RETURN VALUE

None.

SEE ALSO

glPrintf, glPutFont, doprnt

void glPrintf(int x, int y, fontInfo *pInfo, char *fmt, ...);

Prints a formatted string (much like **printf**) on the LCD screen. Only the character codes that exist in the font set are printed, all others are skipped. For example, '\b', '\t', '\n' and '\r' (ASCII backspace, tab, new line, and carriage return, respectively) will be printed if they exist in the font set, but will not have any effect as control characters. Any portion of the bitmap character that is outside the LCD display area will be clipped.

PARAMETERS

x is the *x* coordinate (column) of the upper left corner of the text.

y is the *y* coordinate (row) of the upper left corner of the text.

pInfo is a pointer to the font descriptor.

***fmt** is a formatted string.

... are formatted string conversion parameter(s).

EXAMPLE

```
glprintf(0,0, &fi12x16, "Test %d\n", count);
```

RETURN VALUE

None.

SEE ALSO

glXFontInit

`void glRight1(int left, int top, int cols, int rows);`

Scrolls byte-aligned window right one pixel, left column is filled by current pixel type (color).

PARAMETERS

`left` is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

`top` is the top left corner of the bitmap.

`cols` is the number of columns in the window, must be evenly divisible by 8, otherwise truncates.

`rows` is the number of rows in the window.

RETURN VALUE

None.

SEE ALSO

`glHScroll, glLeft1`

`void glUp1(int left, int top, int cols, int rows);`

Scrolls byte-aligned window up one pixel, bottom column is filled by current pixel type (color).

PARAMETERS

`left` is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

`top` is the top left corner of the bitmap.

`cols` is the number of columns in the window, must be evenly divisible by 8, otherwise truncates.

`rows` is the number of rows in the window.

RETURN VALUE

None.

SEE ALSO

`glVScroll, glDown1`

`void glDown1(int left, int top, int cols, int rows);`

Scrolls byte-aligned window down one pixel, top column is filled by current pixel type (color).

PARAMETERS

`left` is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

`top` is the top left corner of the bitmap.

`cols` is the number of columns in the window, must be evenly divisible by 8, otherwise truncates.

`rows` is the number of rows in the window.

RETURN VALUE

None.

SEE ALSO

`glVScroll, glUp1`

`void keypadDef();`

Configures the physical layout of the keypad with the desired ASCII return key codes.

Keypad physical mapping 1 x 7

0	4	1	5	2	6	3
['L']		['U']		['D']		['R']
	['–']		['+']		['E']	

where
- 'D' represents Down Scroll
- 'U' represents Up Scroll
- 'R' represents Right Scroll
- 'L' represents Left Scroll
- '–' represents Page Down
- '+' represents Page Up
- 'E' represents the ENTER key

Example: Do the following for the above physical vs. ASCII return key codes.

```
keyConfig (  3,'R',0, 0, 0, 0, 0 );
keyConfig (  6,'E',0, 0, 0, 0, 0 );
keyConfig (  2,'D',0, 0, 0, 0, 0 );
keyConfig (  4,'-',0, 0, 0, 0, 0 );
keyConfig (  1,'U',0, 0, 0, 0, 0 );
keyConfig (  5,'+',0, 0, 0, 0, 0 );
keyConfig (  0,'L',0, 0, 0, 0, 0 );
```

Characters are returned upon keypress with no repeat.

RETURN VALUE

None.

SEE ALSO

`keyConfig, keyGet, keyProcess`

`void keyScan(char *pcKeys);`

Writes "1" to each row and reads the value. The position of a keypress is indicated by a zero value in a bit position.

PARAMETER

`pcKeys` is a pointer to the address of the value read.

RETURN VALUE

None.

SEE ALSO

`keyConfig, keyGet, keypadDef, keyProcess`

A lithium battery with a nominal voltage of 3 V and a minimum capacity of 165 mA·h is recommended. A lithium battery is strongly recommended because of its nearly constant nominal voltage over most of its life.

The drain on the battery by the RCM3600 is typically 6 µA when no other power is supplied. If a 235 mA·h battery is used, the battery can last about 4.5 years:

$$\frac{235\ \text{mA}\cdot\text{h}}{6\ \mu\text{A}} = 4.5\ \text{years.}$$

The actual life in your application will depend on the current drawn by components not on the RCM3600 and the storage capacity of the battery. The RCM3600 does not drain the battery while it is powered up normally.

Cycle the main power off/on on the RCM3600 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 RCM3600 experience a loss of main power.

> **NOTE:** Remember to cycle the main power off/on any time the RCM3600 is removed from the Protoyping Board or motherboard since that is where the backup battery would be located.

D.1.2 Reset Generator

The RCM3600 uses a reset generator to reset the Rabbit 3000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The reset occurs between 2.55 V and 2.70 V, typically 2.63 V.

The RCM3600 has a reset pin, pin 36 on header J1. This pin provides access to the reset output from the reset generator, and is also connected to the reset input of the Rabbit 3000 to allow you to reset the microprocessor externally. A resistor divider consisting of R21 and R22 attenuates the signal associated with an externally applied reset to prevent it from affecting the reset generator.