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	29.4MHz
Flash Size	256KB
RAM Size	128KB
Connector Type	2 IDC Headers 2x17
Size / Dimension	1.85" x 1.65" (47mm x 42mm)
Operating Temperature	-40°C ~ 85°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/101-0518

TABLE OF CONTENTS

Chapter 1. Introduction 1

1.1 RCM3100 Features 1
1.2 Advantages of the RCM3100 3
1.3 Development and Evaluation Tools 3
1.4 How to Use This Manual 3
1.4.1 Additional Product Information 3
1.4.2 Online Documentation 3

Chapter 2. Hardware Setup 5

2.1 Development Kit Contents 5
2.2 Hardware Connections 6
2.2.1 Attach Module to Prototyping Board 6
2.2.2 Connect Programming Cable 7
2.2.3 Connect Power 8
2.2.3.1 Overseas Development Kits 8
2.3 Starting Dynamic C 9
2.4 Run a Sample Program 9
2.4.1 Troubleshooting 9
2.5 Where Do I Go From Here? 10
2.5.1 Technical Support 10

Chapter 3. Running Sample Programs 11

3.1 Introduction 11
3.2 Sample Programs 12
3.2.1 Serial Communication 13
3.2.2 Real-Time Clock 15
3.2.3 Other Sample Programs 15

Chapter 4. Hardware Reference 17

4.1 RCM3100 Digital Inputs and Outputs 18
4.1.1 Memory I/O Interface 23
4.1.2 Other Inputs and Outputs 23
4.1.3 5 V Tolerant Inputs 23
4.2 Serial Communication 24
4.2.1 Serial Ports 24
4.2.2 Serial Programming Port 24
4.3 Serial Programming Cable 25
4.3.1 Changing Between Program Mode and Run Mode 25
4.3.2 Standalone Operation of the RCM3100 26
4.4 Other Hardware 27
4.4.1 Clock Doubler 27
4.4.2 Spectrum Spreader 27
4.5 Memory 28
4.5.1 SRAM 28
4.5.2 Flash EPROM 28
4.5.3 Dynamic C BIOS Source Files 28

Chapter 5. Software Reference 29
5.1 More About Dynamic C ... 29
5.2 Dynamic C Function Calls ... 31
5.2.1 I/O ... 31
5.2.2 Serial Communication Drivers ... 31
5.2.3 Prototyping Board Functions ... 31
5.2.3.1 Board Initialization ... 31
5.3 Upgrading Dynamic C ... 32
5.3.1 Upgrades ... 32

Appendix A. RabbitCore RCM3100 Specifications 33
A.1 Electrical and Mechanical Characteristics ... 34
A.1.1 Exclusion Zone ... 36
A.1.2 Headers ... 37
A.1.3 Physical Mounting ... 37
A.2 Bus Loading ... 38
A.3 Rabbit 3000 DC Characteristics ... 41
A.4 I/O Buffer Sourcing and Sinking Limit ... 42
A.5 Conformal Coating ... 43
A.6 Jumper Configurations ... 44

Appendix B. Prototyping Board 45
B.1 Introduction ... 46
B.1.1 Prototyping Board Features ... 47
B.2 Mechanical Dimensions and Layout ... 49
B.3 Power Supply ... 50
B.4 Using the Prototyping Board ... 51
B.4.1 Adding Other Components ... 52
B.4.2 Measuring Current Draw ... 52
B.4.3 Other Prototyping Board Modules and Options ... 53
B.5 Use of Rabbit 3000 Parallel Ports ... 54

Appendix C. LCD/Keypad Module 57
C.1 Specifications ... 57
C.2 Contrast Adjustments for All Boards ... 59
C.3 Keypad Labeling ... 60
C.4 Header Pinouts ... 61
C.4.1 I/O Address Assignments ... 61
C.5 Mounting LCD/Keypad Module on the Prototyping Board ... 62
C.6 Bezel-Mount Installation ... 63
C.6.1 Connect the LCD/Keypad Module to Your Prototyping Board ... 65
C.7 LCD/Keypad Module Function Calls ... 66
C.7.1 LCD/Keypad Module Initialization ... 66
C.7.2 LEDs ... 66
C.7.3 LCD Display ... 67
C.7.4 Keypad ... 82
C.8 Sample Programs ... 85

Appendix D. Power Supply 87
D.1 Power Supplies ... 87
D.1.1 Battery-Backup Circuits ... 87
D.1.2 Reset Generator ... 88

2.2 Hardware Connections

2.2.1 Attach Module to Prototyping Board

Turn the RCM3100 module so that the mounting holes on the RCM3100 and on the Prototyping Board line up, as shown in Figure 1 below. Align the pins from headers J1 and J2 on the bottom side of the module into header sockets RCM2JA and RCM2JB on the Prototyping Board (these sockets were labeled J12 and J13 on earlier versions of the Prototyping Board).

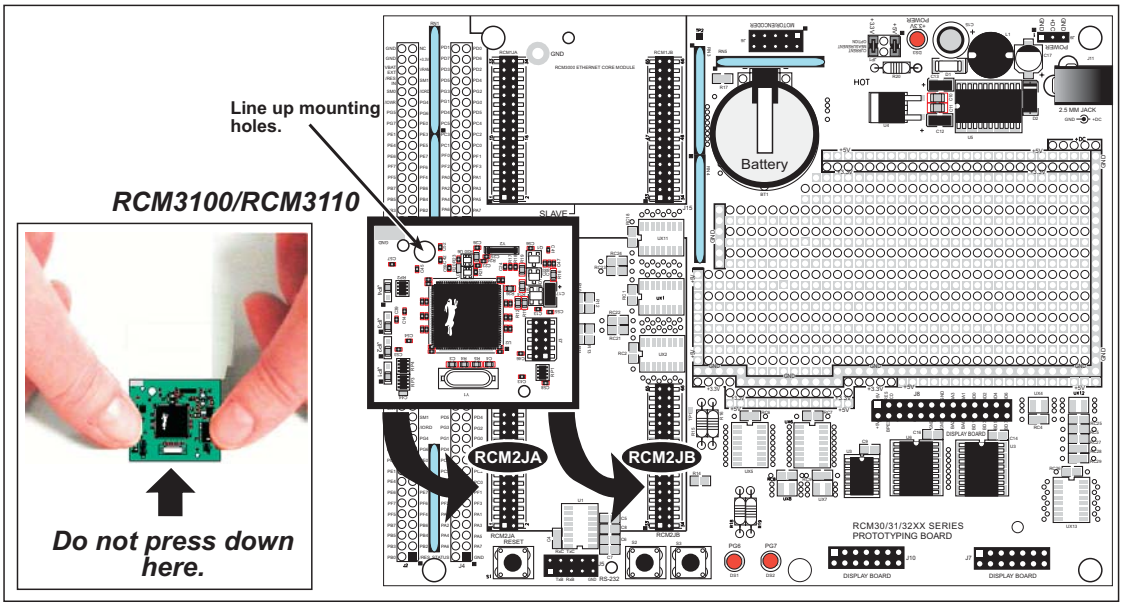

Figure 1. Installing the RCM3100 Module on the Prototyping Board

Although you can install a single module into either the **MASTER** or the **SLAVE** position on the Prototyping Board, all the Prototyping Board features (switches, LEDs, serial port drivers, etc.) are connected to the **MASTER** position — install a single module in the **MASTER** position.

> **NOTE:** It is important that you line up the pins on headers J1 and J2 of the RCM3100 module exactly with the corresponding pins of header sockets RCM2JA and RCM2JB 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 header sockets—press down in the area above the header pins using your thumbs or fingers over the connectors as shown in Figure 1. Do ***not*** press down on the middle of the RCM3100 module to avoid flexing the module, which could damage the module or the components on the module.

Should you need to remove the RCM3100 module, grasp it with your fingers along the sides by the connectors and gently work the module up to pull the pins away from the sockets where they are installed. Do ***not*** remove the module by grasping it at the top and bottom.

4. HARDWARE REFERENCE

Chapter 2 describes the hardware components and principal hardware subsystems of the RCM3100. Appendix A, "RabbitCore RCM3100 Specifications," provides complete physical and electrical specifications.

Figure 4 shows these Rabbit-based subsystems designed into the RCM3100.

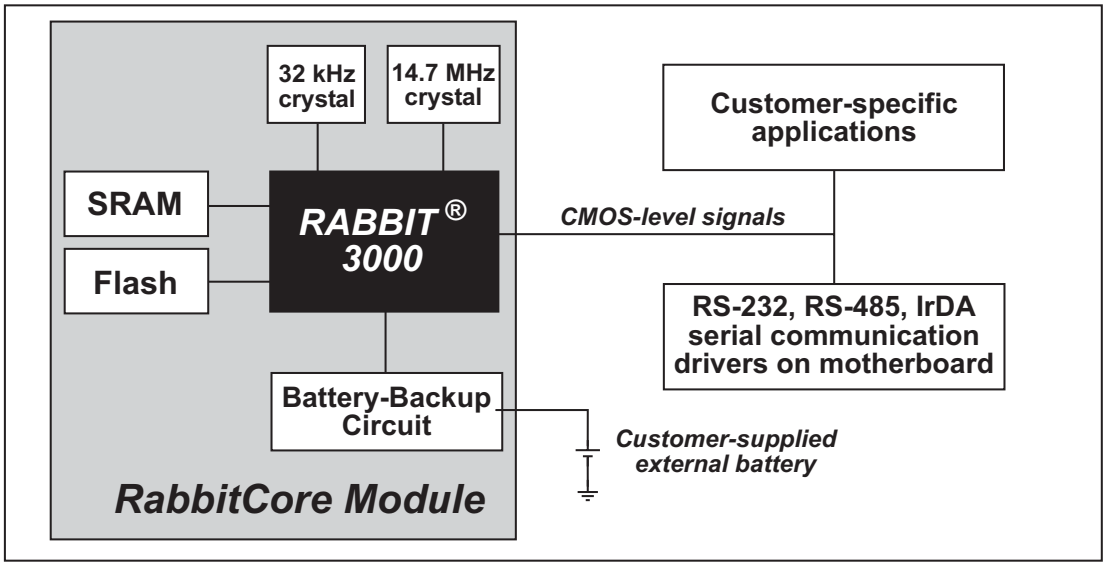

Figure 4. RCM3100 Subsystems

Table 2. RCM3100 Pinout Configurations (continued)

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J2	1	/RES	Reset output	Reset input	Reset output from Reset Generator
	2	PB0	Input/Output	CLKB	
	3	PB2	Input/Output	IA0 /SWR	External Address 0 Slave port write
	4	PB3	Input/Output	IA1 /SRD	External Address 1 Slave port read
	5	PB4	Input/Output	IA2 SA0	External Address 2 Slave port Address 0
	6	PB5	Input/Output	IA3 SA1	External Address 3 Slave port Address 1
	7	PB6	Input/Output	IA4	External Address 4
	8	PB7	Input/Output	IA5 /SLAVEATTN	External Address 5 Slave Attention
	9	PF4	Input/Output	AQD1B PWM0	
	10	PF5	Input/Output	AQD1A PWM1	
	11	PF6	Input/Output	AQD2B PWM2	
	12	PF7	Input/Output	AQD2A PWM3	
	13	PE7	Input/Output	I7 /SCS	
	14	PE6	Input/Output	I6	
	15	PE5	Input/Output	I5 INT1B	
	16	PE4	Input/Output	I4 INT0B	
	17	PE3	Input/Output	I3	
	18	PE1	Input/Output	I1 INT1A	I/O Strobe 1 Interrupt 1A
	19	PE0	Input/Output	I0 INT0A	I/O Strobe 0 Interrupt 0A

4.1.1 Memory I/O Interface

The Rabbit 3000 address lines (A0–A19) 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–PB7 can also be used as an external address bus.

When using the auxiliary I/O bus instead of the default address bus, you must add the following line at the beginning of your program.

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

The STATUS output has three different programmable functions:

5. It can be driven low on the first op code fetch cycle.
6. It can be driven low during an interrupt acknowledge cycle.
7. It can also serve as a general-purpose output.

4.1.2 Other Inputs and Outputs

Two status mode pins, SMODE0 and SMODE1, are available as inputs. The logic state of these two pins determines the startup procedure after a reset.

/RESET_IN is an external input used to reset the Rabbit 3000 microprocessor and the RabbitCore RCM3100 memory. /RES is an output from the reset circuitry that can be used to reset other peripheral devices.

4.1.3 5 V Tolerant Inputs

The RCM3100 operates over a voltage from 3.15 V to 3.45 V, but most RCM3100 input pins, except /RESET_IN, VRAM, VBAT_EXT, and the power-supply pins, are 5 V tolerant. When a 5 V signal is applied to 5 V tolerant pins, they present a high impedance even if the Rabbit power is off. The 5 V tolerant feature allows 5 V devices that have a suitable switching threshold to be connected directly to the RCM3100. This includes HCT family parts operated at 5 V that have an input threshold between 0.8 and 2 V.

> **NOTE:** CMOS devices operated at 5 V that have a threshold at 2.5 V are not suitable for direct connection because the Rabbit 3000 outputs do not rise above VDD, and is often specified as 3.3 V. Although a CMOS input with a 2.5 V threshold may switch at 3.3 V, it will consume excessive current and switch slowly.

In order to translate between 5 V and 3.3 V, HCT family parts powered from 5 V can be used, and are often the best solution. There is also the “LVT” family of parts that operate from 2.0 V to 3.3 V, but that have 5 V tolerant inputs and are available from many suppliers. True level-translating parts are available with separate 3.3 V and 5 V supply pins, but these parts are not usually needed, and have design traps involving power sequencing.

APPENDIX A. RABBITCORE RCM3100 SPECIFICATIONS

Appendix A provides the specifications for the RCM3100, and describes the conformal coating.

A.6 Jumper Configurations

Figure A-6 shows the header locations used to configure the various RCM3100 options via jumpers.

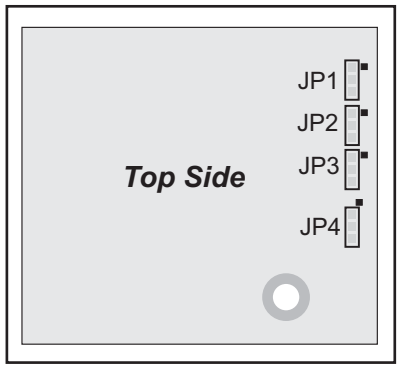

Figure A-6. Location of RCM3100 Configurable Positions

Table A-7 lists the configuration options.

Table A-7. RCM3100 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	Flash Memory Bank Select	1–2	Normal Mode	×
		2–3	Bank Mode	
JP2	Flash Memory Size	1–2	128K/256K	×
		2–3	512K	
JP3	Flash Memory Size	1–2	128K/256K	RCM3100
		2–3	512K	
JP4	SRAM Size	1–2	128K	RCM3110
		2–3	512K	RCM3100

NOTE: The jumper connections are made using 0 Ω surface-mounted resistors.

B.4.1 Adding Other Components

There are pads that can be used for surface-mount prototyping involving SOIC devices. There is provision for seven 16-pin devices (six on one side, one on the other side). There are 10 sets of pads that can be used for 3- to 6-pin SOT23 packages. There are also pads that can be used for SMT resistors and capacitors in an 0805 SMT package. Each component has every one of its pin pads connected to a hole in which a 30 AWG wire can be soldered (standard wire wrap wire can be soldered in for point-to-point wiring on the Prototyping Board). Because the traces are very thin, carefully determine which set of holes is connected to which surface-mount pad.

B.4.2 Measuring Current Draw

The Prototyping Board has a current-measurement feature available on header JP1. Normally, a jumper connects pins 1–2 and pins 5–6 on header JP1, which provide jumper connections for the +5 V and the +3.3 V regulated voltages respectively. You may remove a jumper and place an ammeter across the pins instead, as shown in the example in Figure B-5, to measure the current being drawn.

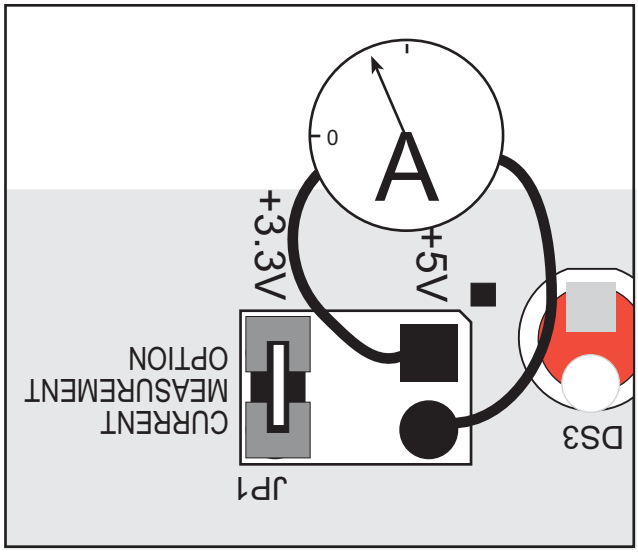

Figure B-5. Prototyping Board Current-Measurement Option

B.4.3 Other Prototyping Board Modules and Options

With the RCM3100 plugged into the **MASTER** slots, it has full access to the RS-232 transceiver, and can act as the "master" relative to another RabbitCore RCM3000, RCM3100, or RCM3200 plugged into the **SLAVE** slots, which acts as the "slave."

An optional LCD/keypad module is available that can be mounted on the Prototyping Board. Refer to Appendix C, "LCD/Keypad Module," for complete information.

The RCM3100 has a 2-channel quadrature decoder and a 10-bit free-running PWM counter with four pulse-width registers. These features allow the RCM3100 to be used in a motor control application, although Rabbit Semiconductor does not offer the drivers or a compatible stepper motor control board at this time.

The Prototyping Board has a header at J6 to which a customer-developed motor encoder may be connected. Figure B-6 shows the motor encoder pinout at header J6.

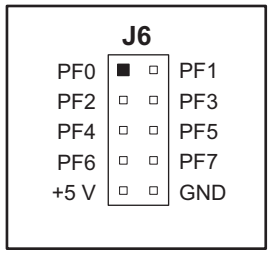

Figure B-6. Prototyping Board Motor Encoder Connector Pinout

Refer to Appendix E, "Motor Control Features," for complete information on using the Rabbit 3000's Parallel Port F in conjunction with this application.

C.7 LCD/Keypad Module Function Calls

When mounted on the Prototyping Board, the LCD/keypad module uses the auxiliary 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 auxiliary I/O bus.

C.7.1 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.7.2 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 **LIB\DISPLAYS\LCD122KEY7.LIB** library.

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

LED on/off control. This function will only work when the LCD/keypad module is installed on the 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 glLeft1(int left, int top, int cols, int rows);

Scrolls byte-aligned window left one pixel, right 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, glRight1

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

C.7.4 Keypad

The functions used to control the keypad are contained in the Dynamic C **LIB\KEYPADS\KEYPAD7.LIB** library.

```
void keyInit(void);
```

Initializes keypad process

RETURN VALUE

None.

SEE ALSO

brdInit

```
void keyConfig(char cRaw, char cPress,
   char cRelease, char cCntHold, char cSpdLo,
   char cCntLo, char cSpdHi);
```

Assigns each key with key press and release codes, and hold and repeat ticks for auto repeat and debouncing.

PARAMETERS

cRaw is a raw key code index.

1x7 keypad matrix with raw key code index assignments (in brackets):

[0]		[1]		[2]		[3]
	[4]		[5]		[6]	

User Keypad Interface

cPress is a key press code

An 8-bit value is returned when a key is pressed.
0 = Unused.

See **keypadDef()** for default press codes.

cRelease is a key release code.

An 8-bit value is returned when a key is pressed.
0 = Unused.

cCntHold is a hold tick.

How long to hold before repeating.
0 = No Repeat.

cSpdLo is a low-speed repeat tick.

How many times to repeat.
0 = None.

cCntLo is a low-speed hold tick.

How long to hold before going to high-speed repeat.
0 = Slow Only.

APPENDIX E. MOTOR CONTROL FEATURES

The RCM30/31/32XX Prototyping Board has a header at J6 for a motor control connection. While Rabbit Semiconductor does not have the drivers or a compatible stepper motor control board at this time, this appendix provides additional information about Parallel Port F on the Rabbit 3000 microprocessor to enable you to develop your own application.

E.1 Overview

The Parallel Port F connector on the Prototyping Board, J6, gives access to all 8 pins of Parallel Port F, along with +5 V. This appendix describes the function of each pin, and the ways they may be used for motion-control applications. It should be read in conjunction with the ***Rabbit 3000 Microprocessor User's Manual*** and the RCM3100 and the RCM3000/RCM3100/RCM3200 Prototyping Board schematics.

Table E-2. Parallel Port F Registers (continued)

Register Name	Mnemonic	I/O Address	R/W	Reset Value
Port F Data Direction Register	PFDDR	00111111 (0x3F)	W	00000000
Bits	**Value**	**Description**		
0:7	0	Corresponding port bit is an input		
	1	Corresponding port bit is an output		

E.4 PWM Outputs

The Pulse-Width Modulator consists of a 10-bit free-running counter and four width registers. Each PWM output is high for $n + 1$ counts out of the 1024-clock count cycle, where n is the value held in the width register. The PWM output high time can optionally be spread throughout the cycle to reduce ripple on the externally filtered PWM output. The PWM is clocked by the output of Timer A9. The spreading function is implemented by dividing each 1024-clock cycle into four quadrants of 256 clocks each. Within each quadrant, the Pulse-Width Modulator uses the eight MSBs of each pulse-width register to select the base width in each of the quadrants. This is the equivalent to dividing the contents of the pulse-width register by four and using this value in each quadrant. To get the exact high time, the Pulse-Width Modulator uses the two LSBs of the pulse-width register to modify the high time in each quadrant according to Table E-3 below. The "n/4" term is the base count, and is formed from the eight MSBs of the pulse-width register.

Table E-3. PWM Outputs

Pulse Width LSBs	1st	2nd	3rd	4th
00	n/4 + 1	n/4	n/4	n/4
01	n/4 + 1	n/4	n/4 + 1	n/4
10	n/4 + 1	n/4 + 1	n/4 + 1	n/4
11	n/4 + 1	n/4 + 1	n/4 + 1	n/4 + 1

The diagram below shows a PWM output for several different width values for both modes of operation. Operation in the spread mode reduces the filtering requirements on the PWM output in most cases.

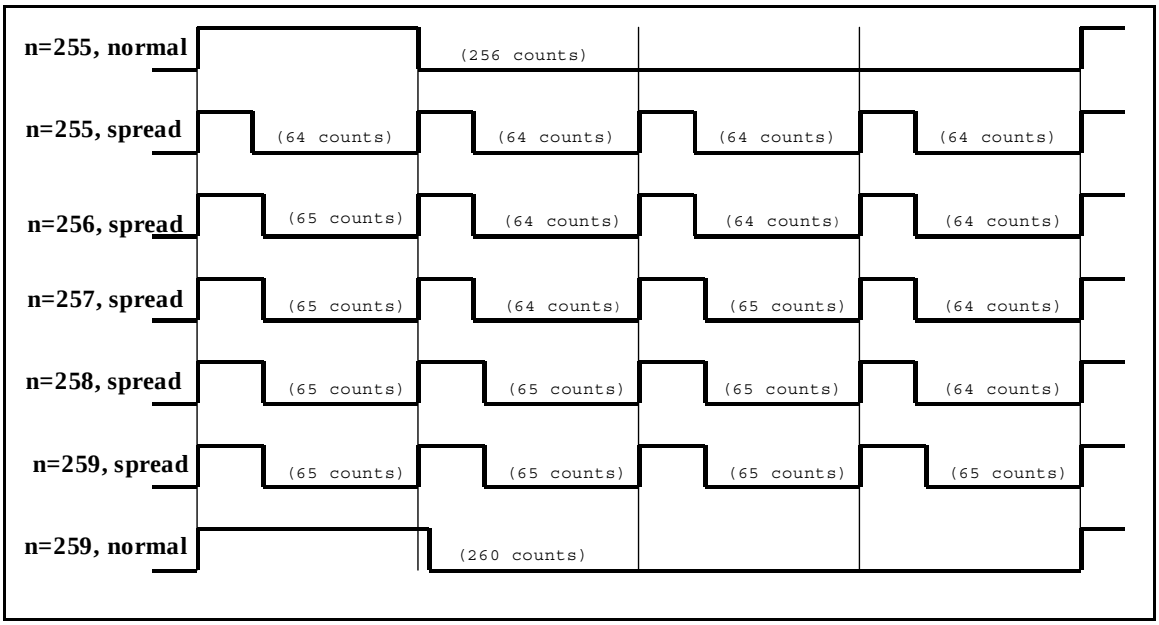

Figure E-1. PWM Outputs for Various Normal and Spread Modes

E.5 PWM Registers

There are no default values on reset for any of the PWM registers.

Table E-4. PWM Registers

PWM LSBs	Register	Address
	PWL0R	10001000 (0x88)
	PWL1R	10001010 (0x8A)
	PWL2R	10001100 (0x8C)
	PWL3R	10001110 (0x8E)
Bit(s)	**Value**	**Description**
7:6	Write	The least significant two bits for the Pulse Width Modulator count are stored
5:1		These bits are ignored.
0	0	PWM output High for single block.
	1	Spread PWM output throughout the cycle
PWM MSB x	**Register**	**Address**
	PWM0R	Address = 10001001 (0x89)
	PWM1R	Address = 10001011 (0x8B)
	PWM2R	Address = 10001101 (0x8D)
	PWM3R	Address = 10001111 (0x8F)
Bit(s)	**Value**	**Description**
7:0	write	The most significant eight bits for the Pulse-Width Modulator count are stored With a count of *n*, the PWM output will be high for *n* +1 clocks out of the 1024 clocks of the PWM counter.

INDEX

A

additional information
online documentation 3
auxiliary I/O bus 23
software 66

B

battery backup
battery life 88
external battery connections 87
real-time clock 88
reset generator 88
board initialization
function calls 31
brdInit 31
bus loading 38

C

clock doubler 27
conformal coating 43

D

Development Kit 5
RCM3100 3
digital I/O 18
I/O buffer sourcing and sinking limits 42
memory interface 23
SMODE0 ..... 23, 24
SMODE1 ..... 23, 24
dimensions
LCD/keypad template 60
Prototyping Board 49
RCM3100 34
Dynamic C 29
add-on modules 32
sample programs 12
standard features 30
debugging 30
telephone-based technical support 32
upgrades and patches 32

E

exclusion zone 36

F

features 1
Prototyping Board 46, 47
flash memory addresses
user blocks 28

H

hardware connections 6
install RCM3100 on Prototyping Board 6
power supply 8
programming cable 7
hardware reset 8

I

I/O address assignments
LCD/keypad module 61
I/O buffer sourcing and sinking limits 42

J

jumper configurations 44
JP1 (flash memory bank select) 28, 44
JP2 (flash memory size) 44
JP3 (flash memory size) 44
JP4 (SRAM size) 44
jumper locations 44

K

keypad template 60
removing and inserting label . 60

L

LCD/keypad module
bezel-mount installation 63
dimensions 58
function calls
dispInit 66
header pinout 61
I/O address assignments ... 61
keypad
function calls
keyConfig 82
keyGet 83
keyInit 82
keypadDef 84
keyProcess 83
keyScan 84
keyUnget 83
keypad template 60
LCD display
function calls
glBackLight 67
glBlankScreen 68
glBlock 68
glBuffLock 74
glBuffUnlock 74
glDispOnOff 67
glDown1 77
glFillCircle 71
glFillPolygon 70
glFillScreen 68
glFillVPolygon 70
glFontCharAddr 71
glGetBrushType 75
glGetPfStep 72
glHScroll 77
glInit 67
glLeft1 76
glPlotCircle 70
glPlotDot 75
glPlotLine 75
glPlotPolygon 69
glPlotVPolygon 69
glPrintf 73
glPutChar 73
glPutFont 72
glRight1 76
glSetBrushType 74
glSetContrast 68