PRODUCT MANUAL

Rabbit® RIO™

User's Manual

019–0158 • 080930–E

Rabbit RIO User's Manual

Part Number 019-0158 • 080930–E • Printed in U.S.A.

Trademarks

Rabbit and Dynamic C are registered trademarks of Digi International Inc.

Rabbit RIO is a trademark of Digi International Inc.

The latest revision of this manual is available on the Rabbit Web site, www.rabbit.com, for free, unregistered download.

Digi International Inc.

www.rabbit.com

TABLE OF CONTENTS

1. THE RABBIT RIO

1.1 Overview

The Rabbit RIO is a peripheral device designed to be incorporated into systems requiring versatile timing controls and a broader range of functionality. The Rabbit RIO can be used with any microprocessor.

The Rabbit RIO communicates with the microprocessor in either a parallel or a serial mode. The particular communication mode is determined during power-up. In the parallel mode, the chip communicates with the microprocessor through a parallel bus with eight data bits, five address bits, and four control bits. The serial mode can be used for bidirectional data flow on one wire or via the SPI and RabbitNet protocols. In the serial mode, the parallel data lines are available to be used as general-purpose I/O. The multiple communication modes allow the Rabbit RIO to be a part of a wide variety of systems that use any one of these communication methods.

Implementing the Rabbit RIO as a RabbitNet hub provides a simple, efficient, and flexible means of establishing a network of RabbitNet peripheral cards. The RabbitNet architecture allows a hub to connect to seven peripheral cards, and support for two levels of hubs allows a master device to control up to 49 RabbitNet peripheral cards.

The design of the Rabbit RIO's I/O blocks allows any of the eight identical I/O blocks, each with four bits or I/O pins, to be programmed to perform any number of different functions, including a pulse-width modulator, a pulse-position modulator, event counters, quadrature decoders, pulse measurements, and I/O, including pin-pair protection for applications such as H-bridge drivers.

The main clock can be used directly by each I/O block, or it may be prescaled down to a lower frequency. Either clock source can be used by the 16-bit counter, which is the core of each I/O block. This counter is complemented by a number of registers that provide access and control to the counter for the various Rabbit RIO functions that it involves.

The Rabbit RIO can be incorporated without any glue logic in a Rabbit-based system, enabling a more efficient use of resources. Rabbit's Dynamic C software allows for seamless integration of hardware and software. Dynamic C provides a complete set of function calls to enable you to use the Rabbit RIO without having to write any additional drivers.

The Rabbit RIO can operate at clock speeds up to 40 MHz. It is powered by 3.3 V, but the I/O are 5 V tolerant. The Rabbit RIO is packaged in a 64-pin 10 mm ×10mm TQFP, making its small footprint and low profile ideal for embedded applications.

1.2 Key Features

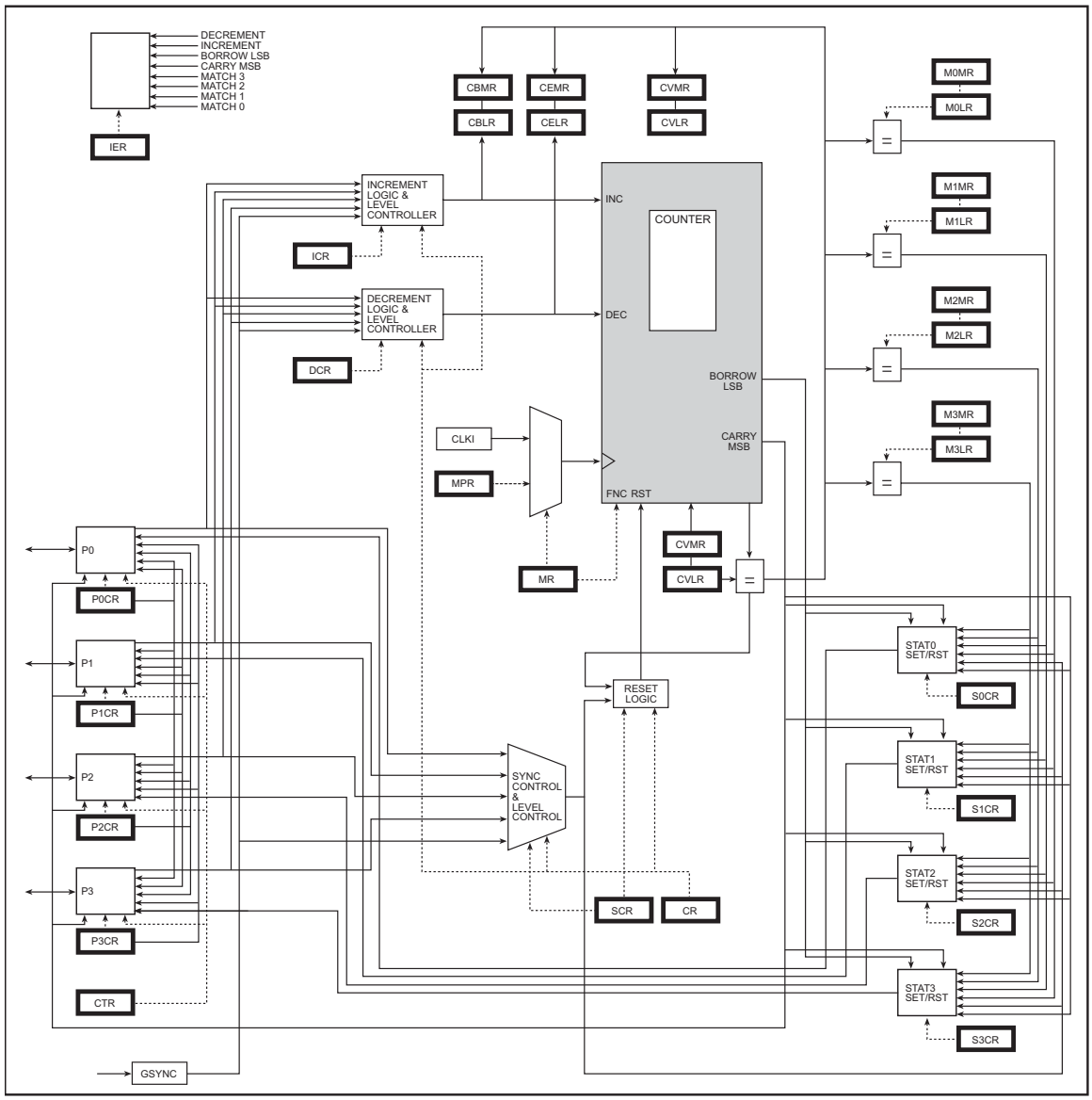
- 5 V tolerant
- Clock speed up to 40 MHz
- 64-pin 10 mm ×10 mm TQFP package
- Multiple communication interfaces — SPI, parallel, and RabbitNet
- 8 independent functional I/O blocks with 4 pins each
- Any pin on each I/O block is capable of:
  - ▶ Generating PWM outputs and variable-phase PWM outputs
  - ▶ Pulse count
  - ▶ Input capture (pulse length or frequency)
  - ▶ Decoding quadrature signals
  - ▶ Provide extended I/O pins to the microprocessor
  - ▶ Pin-pair protection for driving H bridges
- Up to 32 digital I/O lines, up to 4 general-purpose inputs
- Global or block synch input to coordinate outputs
- Interrupt request pin
- RoHS compliant
- High-performance 8-bit device requires no glue logic to Rabbit systems
- RabbitNet hub feature allows control of up to seven RabbitNet devices in each of two levels for a total of up to 49 RabbitNet devices
- Functionality well-suited for machine control
- Dynamic C libraries allow for Rabbit RIO to be up and running in no time
- Small footprint and multiple functions allow for versatile system

1.3 Development and Evaluation Tools

Rabbit also has an application kit featuring the Rabbit RIO to provide the harware and software tools to help you use the Rabbit RIO for I/O expansion.

- RIO Programmable I/O Kit [Part No. 101-1147 (North American markets) and Part No. 101-1148 (overseas markets)]—comes with two CD-ROMs that includes Dynamic C 10.11 or a later version, an RCM4110 RabbitCore module, and a RIO Prototyping Board. The software bundle on the supplemental CD provides the Dynamic C function calls and sample programs that illustrate the use of the Rabbit RIO chip included on the RIO Prototyping Board and can serve as a template for you to develop your own application.

1.4 Block Diagram of Rabbit RIO I/O Blocks



1.5 Pin Functions and Descriptions

Pin Group	Pin Name	Direction	Function
Hardware	/RESET	Input	Master Reset
	CLK	Input	Clock In
CPU Buses	BLOCK[2:1] or GPIN[2:1] BLOCK[0] G//B /P/I	Input	Address Bus or GPI*
	D7/SERCLK D6/SERI D5/BL6Pin[3] D4/BL6Pin[2] D3/BL6Pin[1] D2/BL7Pin[3] D1/BL7Pin[2] D0/BL7Pin[1]	Bidirectional	Parallel Data Bus or Serial Control Bus & I/O Block Pins
Status & Control	/CS	Input	I/O Chip Select
	/IORD or GPIN[4]	Input	I/O Read Enable or GPI*
	/IOWR or GPIN[3]	Input	I/O Write Enable or GPI*
	/INT	Output	Interrupt Request
	/WAIT/SERO	Output	Wait Request or Serial Out
	SER//PAR	Input	Serial/Parallel Bus Select
Shared	GSYNC	Input	Global Sync
I/O Pins	BL0Pin[3:0] – BL5Pin[3:0] & BL6Pin[0] & BL7Pin[0]	Input/Output	I/O Block Pins
Power	VDDINT VDDIO	Power	Internal Power I/O Power
Ground	VSSINT VSSIO	Ground	Internal Ground I/O Ground

* The GPI options are general-purpose inputs when operating in a serial mode.

1.6 Pinouts

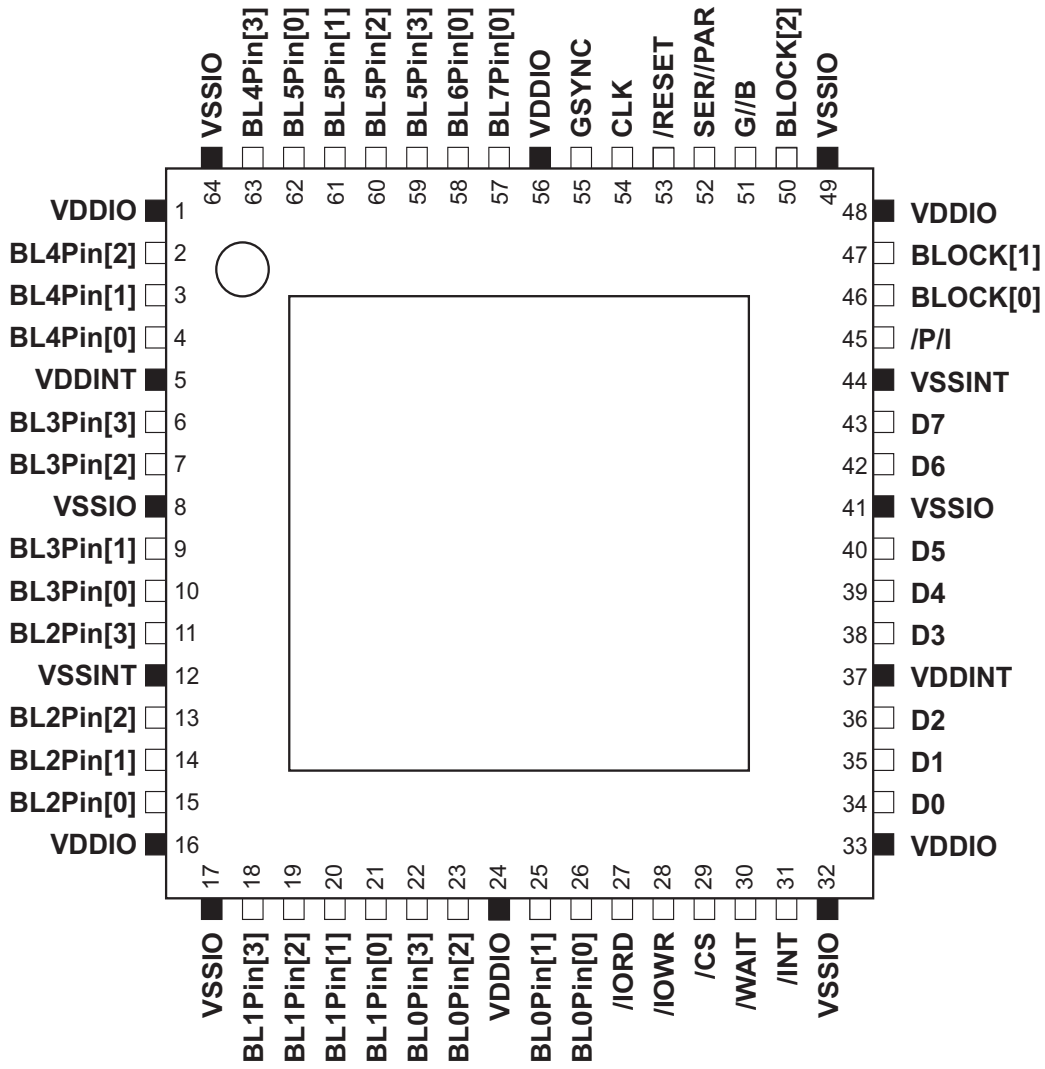


Figure 1-1. Parallel Pinout

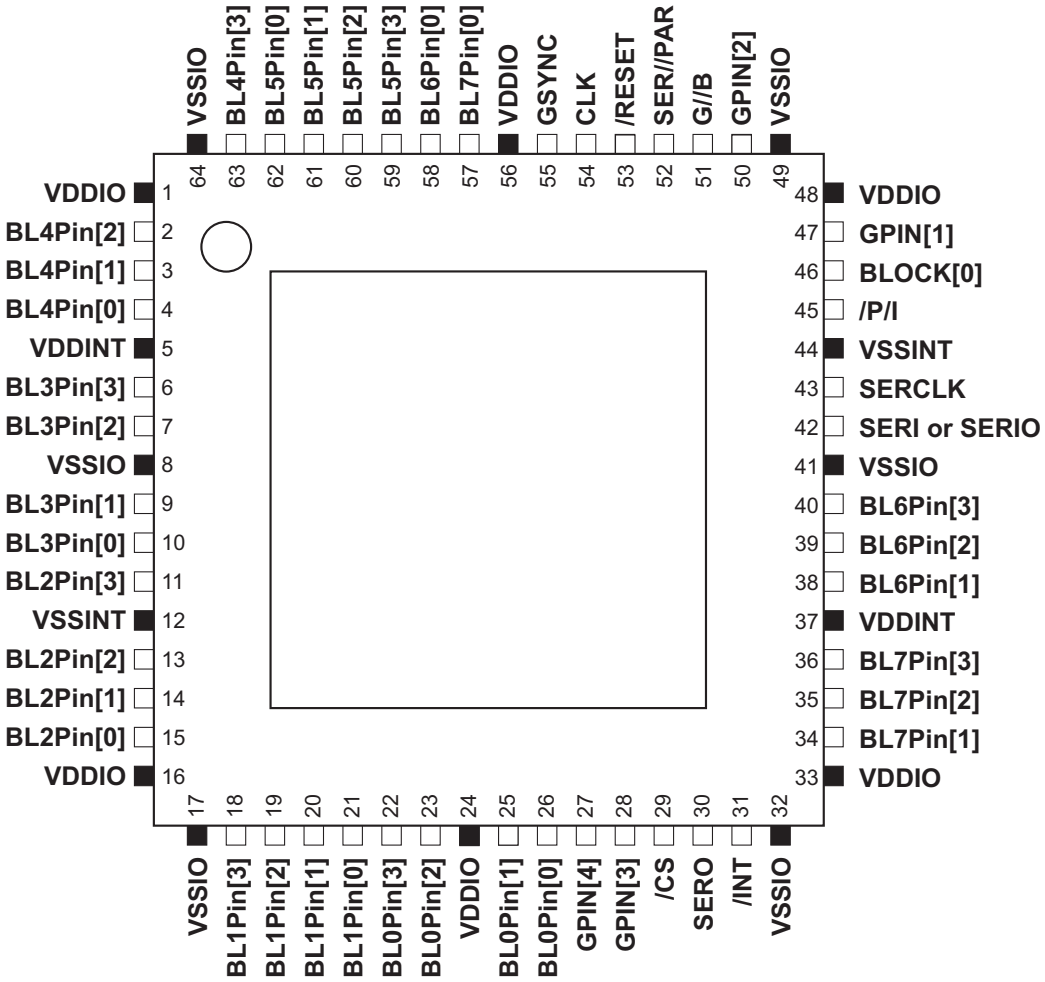


Figure 1-2. Serial Pinout — SPI Interface Mode

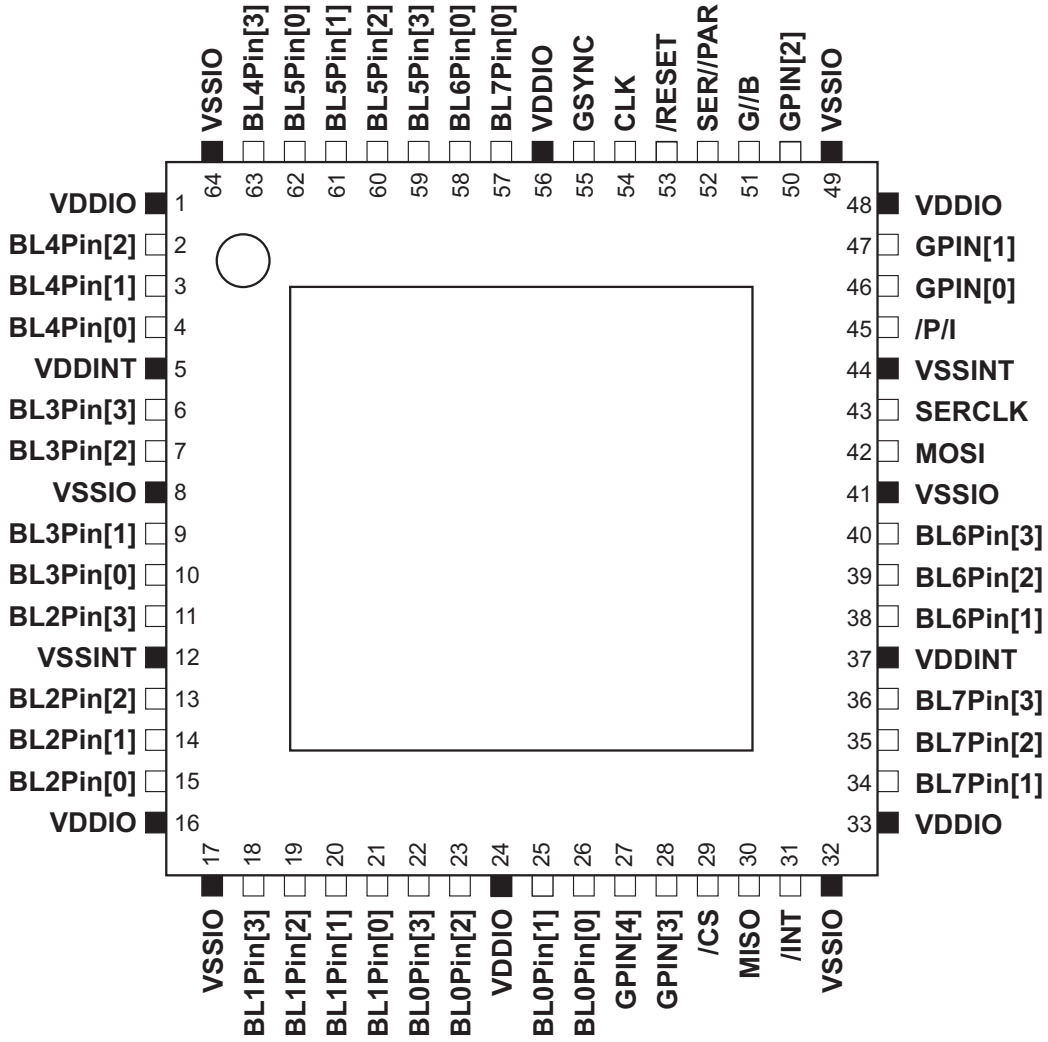


Figure 1-3. Serial Pinout — RabbitNet Device Interface Mode

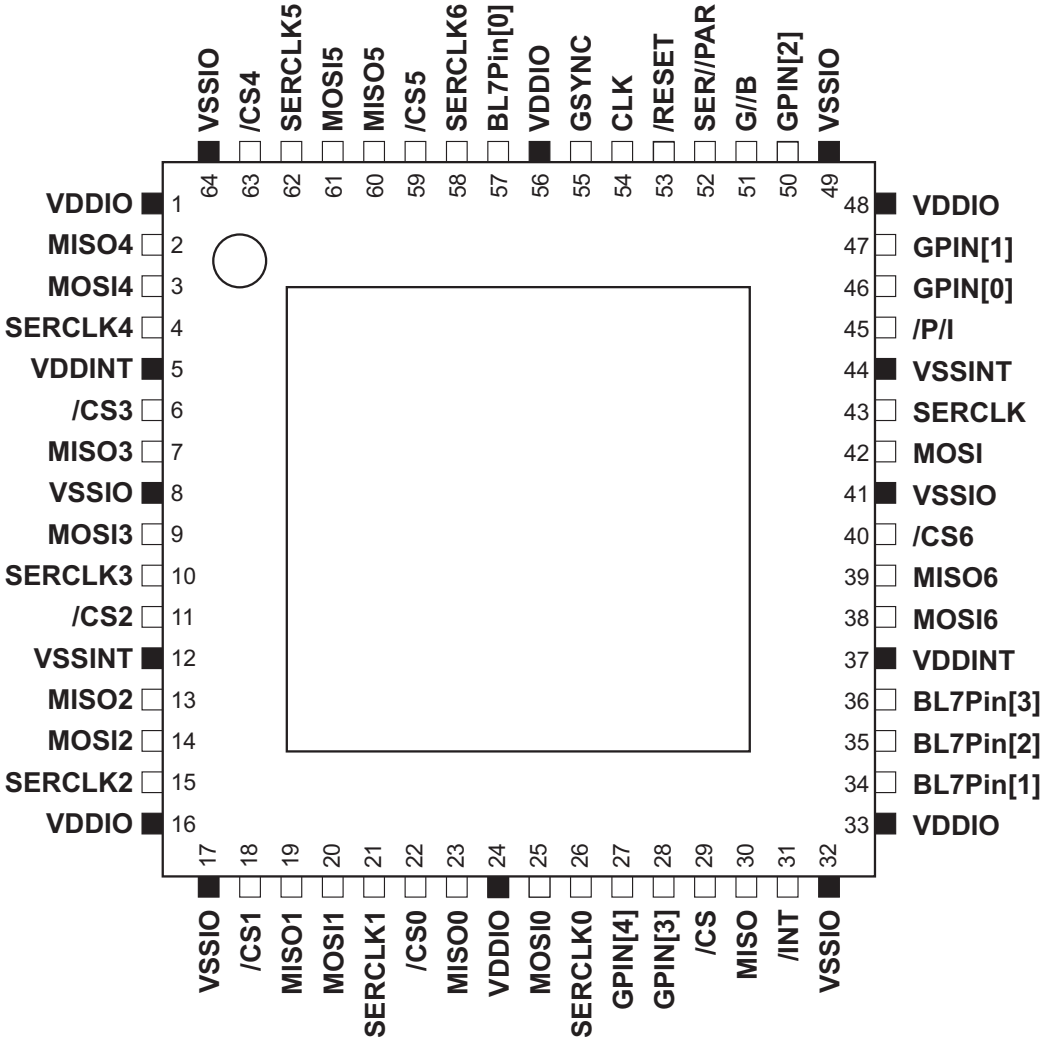


Figure 1-4. Serial Pinout — RabbitNet Hub Interface Mode

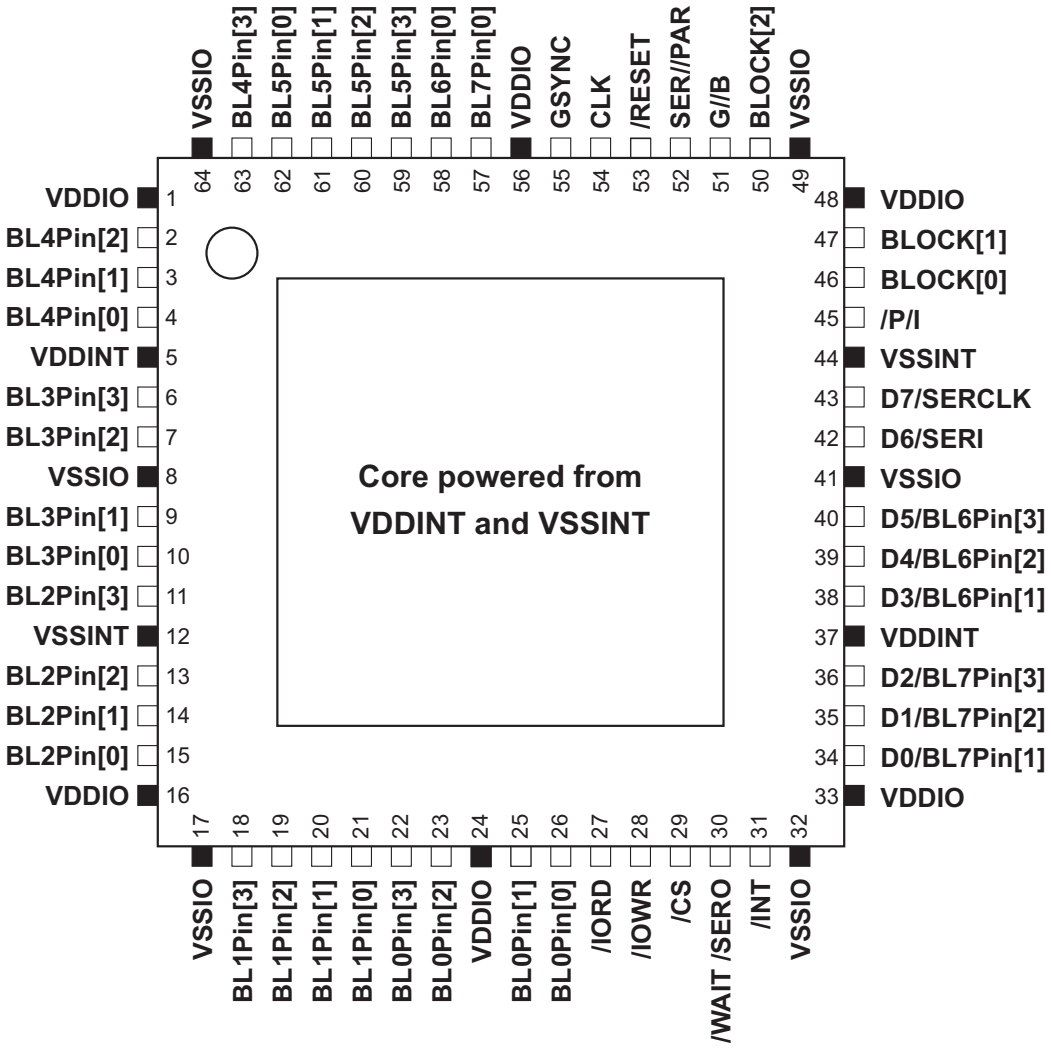


Figure 1-5. General Pinout

1.7 Mechanical Dimensions and Land Pattern — TQFP Package

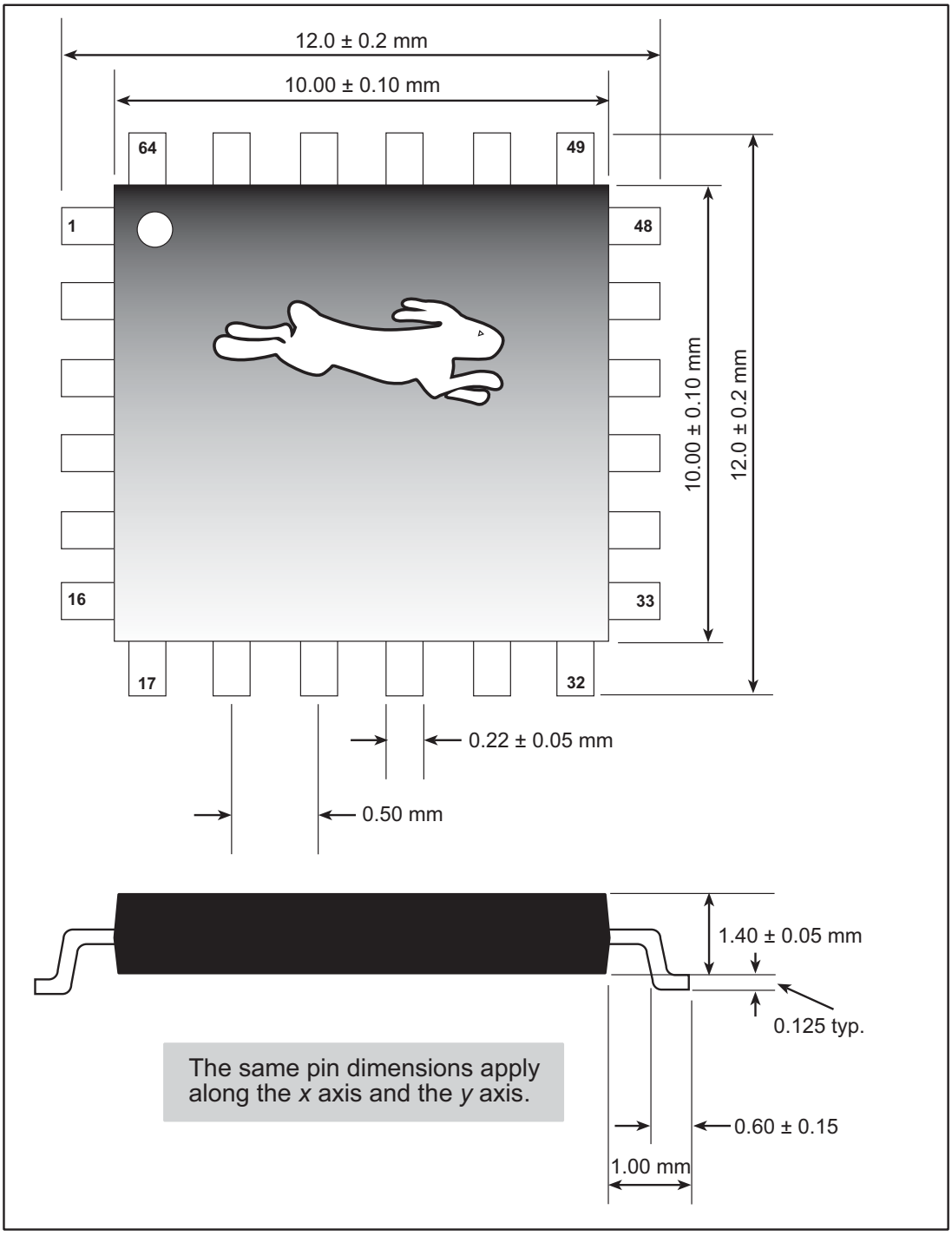


Figure 1-6. Mechanical Dimensions Rabbit RIO TQFP Package

Figure 1-7 shows the PC board land pattern for the Rabbit RIO in a 64-pin TQFP package. This land pattern is based on the IPC-SM-782 standard developed by the Surface Mount Land Patterns Committee and specified in ***Surface Mount Design and Land Pattern Standard***, IPC, Northbrook, IL, 1999.

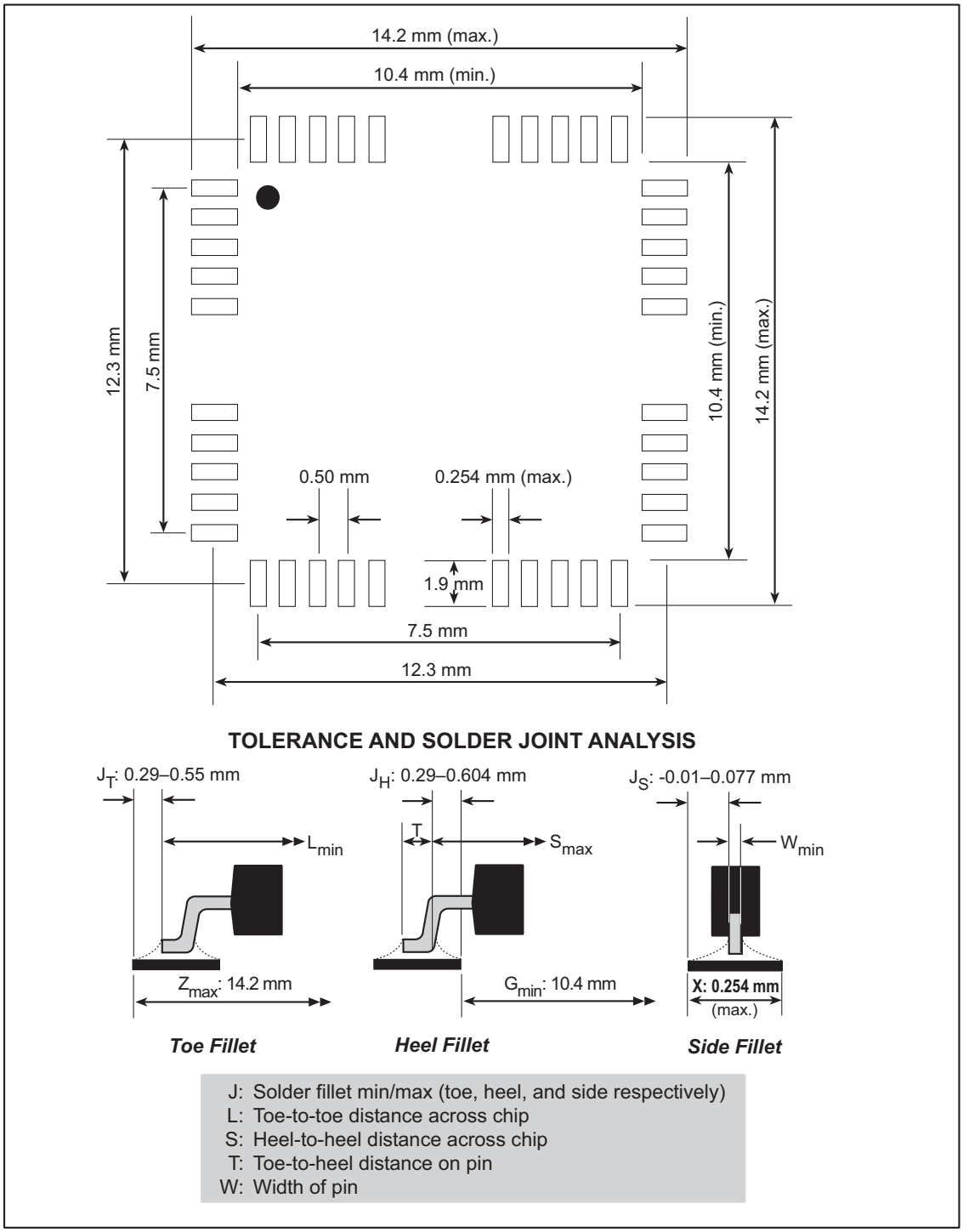


Figure 1-7. PC Board Land Pattern for Rabbit RIO 64-pin TQFP

1.8 DC Characteristics

Table 1-1. Preliminary DC Electrical Characteristics

Parameter		Symbol	Min	Typ	Max
Operating Temperature		T_A	-40°C		85°C
Storage Temperature			-55°C		125°C
Core	Core Supply Voltage	VDD_{CORE}	3.0 V	3.3 V	3.6 V
	Core Current @ 22.1184 MHz, 25°C	I_{CORE}		31.3 mA	
	Core current @ 11.0592 MHz, 25°C			16.3 mA	
	Core current @ 7.3728 MHz, 25°C			11.0 mA	
	Core current @ 3.6864 MHz, 25°C			5.5 mA	
I/O Ring	I/O Ring Supply Voltage	VDD_{IO}	3.0 V	3.3 V	3.6 V
	I/O Ring Current @ 22.1184 MHz, 25°C	I_{IO}		1.1 mA	
	I/O Ring Current @ 11.0592 MHz, 25°C	I_{IO}		1.0 mA	
	I/O Ring Current @ 7.3728 MHz, 25°C			0.9 mA	
	I/O Ring Current @ 3.6864 MHz, 25°C			0.9 mA	
	Input Low Voltage (VDD_{IO} = 3.3 V)	V_{IL}		0.8 V	
	Input High Voltage (VDD_{IO} = 3.3 V)	V_{IH}		2.0 V	
	Output Low Voltage (VDD_{IO} = 3.3 V)	V_{OL}		0.4 V	
	Output High Voltage (VDD_{IO} = 3.3 V)	V_{OH}		2.4 V	
	Maximum I/O Input Voltage			3.3 V	5.0 V
	Output Drive	I_{DRIVE}			8 mA

1.9 AC Characteristics

Table 1-2. Preliminary AC Electrical Characteristics

Parameter	Symbol	Min	Typ	Max
Clock Frequency	f_{main}			40 MHz

1.10 Memory Access Times

All access time measurements are taken at 50% of the signal height.

1.10.1 Parallel Mode

Table 1-3. Parallel Bus Read Time Delays (VDD = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Min	Typ	Max
Clock to Address Delay	T_{adr}			6 ns
Clock to Chip Select Delay	T_{IOCS}			6 ns
Clock to Output Enable Delay	T_{IORD}			6 ns
Data Setup Time	T_{setup}	1 ns		
Data Hold Time	T_{hold}	0 ns		

Table 1-4. Parallel Bus Write Time Delays (VDD = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Min	Typ	Max
Clock to Address Delay	T_{adr}			6 ns
Clock to Chip Select Delay	T_{IOCS}			6 ns
Clock to Write Strobe Delay	T_{IOWR}			6 ns
High Z to Data Valid Relative to Clock	T_{DHZV}			10 ns
Data Valid to High Z Relative to Clock	T_{DVHZ}			10 ns

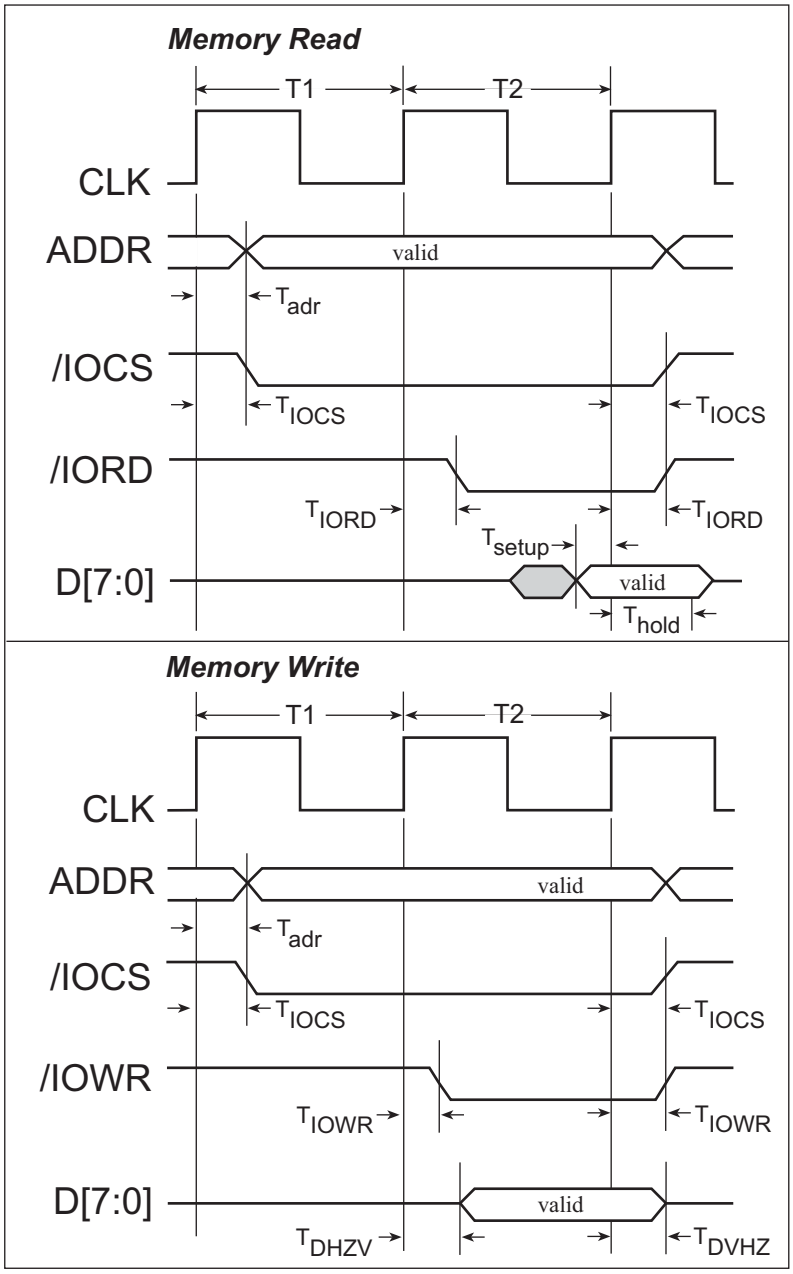


Figure 1-8. Memory Read and Write Cycles

1.10.2 SPI/RabbitNet Mode

Table 1-5. SPI/RabbitNet Bus Access Time Delays (VDD = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Min	Typ	Max
Serial Clock Period	T_{CP}	25 ns		
Serial Clock Pulse Width High	T_{CH}	12 ns		
Serial Clock Pulse Width Low	T_{CL}	12 ns		
Chip Select Fall to Input Data Valid	T_{CSD}	0 ns		
Chip Select Leading Time Before First Clock Edge	T_L	25 ns		
Chip Select Trailing Timer After Last Clode Edge	T_T	25 ns		
Input Data Setup Time	T_{DS}	12 ns		
Input/Output Data Hold Time	T_{DH}	12 ns		
Chip Select Pulse High	T_{CSW}	100 ns		

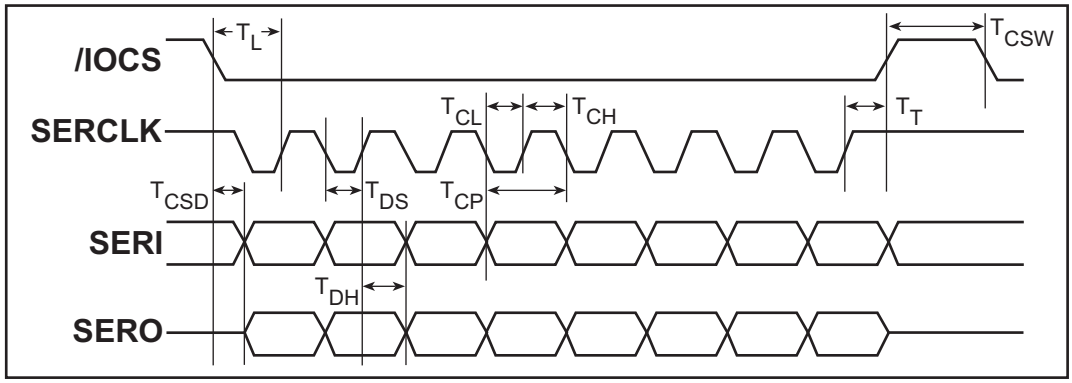


Figure 1-9. SPI/RabbitNet Bus Access Time Delays

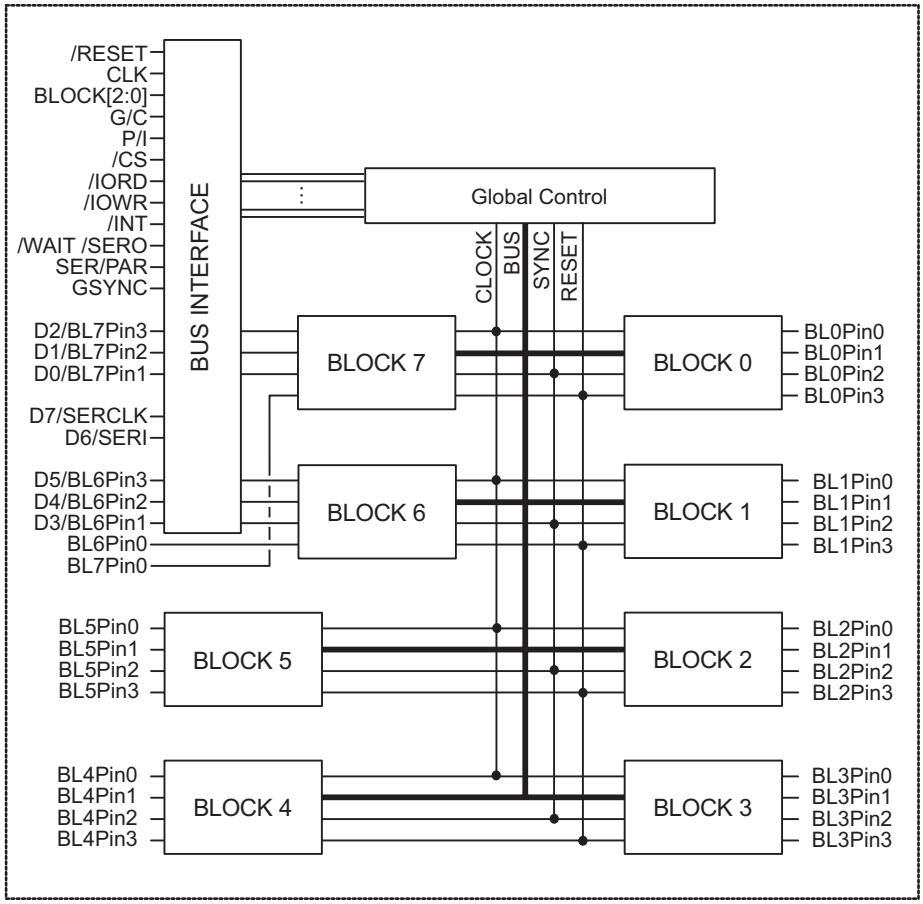
2. MASTER-LEVEL FEATURES

2.1 Overview

The Rabbit RIO uses master-level registers to access individual I/O blocks or to control their overall function. Unlike block registers, master registers can be accessed directly through the five-bit address and eight-bit data bus. In the parallel communication mode, these two buses are accessed through address and data bus pins. In the serial communication mode, the data and address must be decoded from the serial bit stream. Changing the communication mode is a simple matter of pulling the G//B, /P/I, BLOCK[0], and SER//PAR pins high or low.

2.2 Block Diagram

Rabbit RIO Chip Master-Level Features



2.3 Clocks

Even though the Rabbit RIO has just one **Master Clock** pin, the timing of each I/O block is still versatile because the **Master Precaler** will allow any 8-bit prescale of the master clock to be used by the I/O blocks. Once the prescale value is set in the **Master Prescale Register (MPR)**, any I/O block can be set to use either the precaled clock or the master clock directly. Resetting bit 3 of the I/O block's **Mode Register (MR)** allows the I/O block to use the master clock; setting bit 3 allows the I/O block to use the prescaler.

2.4 Reset

A reset signal can be triggered from multiple sources. A *hardware reset* is generated by pulling the reset pin low. This will cause all the master and block-level registers to go back to their original startup state. Section 3.2 lists the reset states for each register. Another method of resetting all the registers is through a *software reset*, which can be accomplished by setting bit 7 of the **Master Control Register (MCR)**. Note that pin-pair protection can only be disabled through a hardware reset.

Additional hardware and software reset options are available at the block level. Such a reset will simply reset the counter for that I/O block, as opposed to resetting the whole chip. The synch signal is used to perform these resets, and there are multiple sources for the synch signal that are specified by the I/O block's **Synch Control Register (SCR)**. This register will specify whether the global synch pin or which one of the four I/O block pins will provide the synch signal.

2.5 Bus Interface

The Rabbit RIO is designed to be connected to the processor data bus, the processor I/O bus, a clocked serial interface, or a RabbitNet master without any glue logic. Unless the Rabbit RIO is acting as a RabbitNet hub, it has all the same functionality regardless of the communication mode — it simply communicates differently in the various communication modes.

When using the parallel communication mode, not all the I/O blocks have complete I/O capability. With this interface, Blocks 6 and 7 each have only one pin available for I/O.

The serial mode is enabled by tying the SER//PAR input *high*. The six possibilities for the serial interface are shown in Table 2-1.

Table 2-1. Serial Mode Interfaces

Pin			Selects	Serial Order
G//B	/P/I	BLOCK[0]		
0	0	X	RabbitNet Device	MSB first
0	1	X	RabbitNet Hub	MSB first
1	0	0	Clocked Serial with Serial Out & Serial In	LSB first
1	0	1	Clocked Serial with Serial Out & Serial In	MSB first
1	1	0	Clocked Serial with Bidirectional Serial I/O	LSB first
1	1	1	Clocked Serial with Bidirectional Serial I/O	MSB first

With a clocked serial interface, data are transferred to and from the Rabbit RIO in address and data cycles. First, the byte of address information is shifted into the chip, and then the byte of data is either shifted in or out of the chip. The active low chip select must remain active for the duration of each byte transfer, but is allowed to go inactive between transfers. The read/write bit is shifted with the address to signal the Rabbit RIO what kind of transfer is requested and sets the direction. Transfers can be either LSB-first or MSB-first. When using the clocked serial interface, all eight I/O blocks have full I/O capabilities.

As a RabbitNet device, the Rabbit RIO implements the RabbitNet specification, including the watchdog function. All RabbitNet transfers are either read-only or are simultaneous read and write (fully duplex). Data are always transferred MSB-first in this mode. When using the Rabbit RIO as a RabbitNet device, all eight I/O blocks have full I/O capabilities.

As a RabbitNet hub, the Rabbit RIO implements the RabbitNet specification for a hub and can be used as either a first- or second-level hub. Only one I/O block has I/O capability, as the pins that are normally used for the other seven I/O blocks are connected to downstream RabbitNet devices.

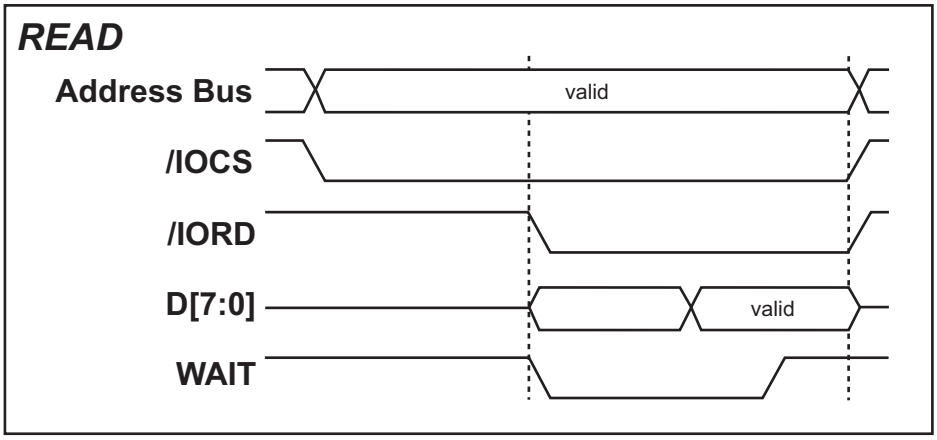
2.5.1 Parallel Mode

The parallel mode is selected by tying the SER//PAR input *low.* This mode is completely asynchronous. This makes the chip useful in systems that do not use Rabbit microprocessors.

The address bus consists of a three-bit I/O block address (BLOCK[2:0]) to select the appropriate I/O block, a Pointer/Indirect (/P/I) bit to select between the two externally addressable registers in each I/O block, and the Global/Block (G//B) bit to select between a global access and a block access.

All external transactions are synchronized internally to the clock, which requires a recovery time of four clock cycles between external accesses. In other words, the maximum rate at which external accesses can occur is once every four internal clock cycles. The /WAIT signal is activated until the Rabbit RIO is ready to accept or supply data during a transaction. The /WAIT signal enforces the recovery time between external transactions, and in the case of a read, is guaranteed to be deasserted once the read data are valid.

A read transaction is shown below. The data bus is driven while /IORD and /CS are both active. The address on the address bus must remain valid for the duration of the transaction, but there is no hold time relative to the trailing edge of /IORD or /CS.



A write transaction is shown below. The data bus is latched at the end of the transaction, with no hold time. As in the case of a read, the address on the address bus must remain valid for the duration of the transaction.

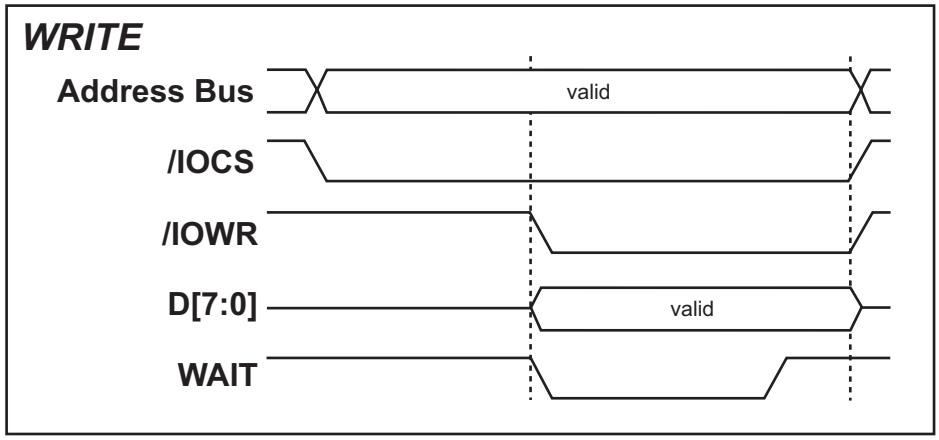


Table 2-2 lists the suggested connections to the Rabbit RIO for the parallel mode using the auxiliary I/O bus.

Table 2-2. Rabbit RIO Connections — Parallel Mode

Rabbit Microprocessor Signal	Rabbit RIO Pin	Description
	SER//PAR	Pulled down
PE1	/CS	Chip select*
PA0	D0	Data bus line
PA1	D1	Data bus line
PA2	D2	Data bus line
PA3	D3	Data bus line
PA4	D4	Data bus line
PA5	D5	Data bus line
PA6	D6	Data bus line
PA7	D7	Data bus line
PB2	/P/I	Pointer/Indirect line
PB3	BLOCK[0]	Block address 0
PB4	BLOCK[1]	Block address 1
PB5	BLOCK[2]	Block address 2
PB7	G//B	G//B (Global/Block) line
PE4	/WAIT	Wait signal
/IORD	/IORD	Read enable
/IOWR	/IOWR	Write enable

* Rabbit RIO /CS may be connected to PE1 or any other available pin on Parallel Port E, or it may be pulled low.

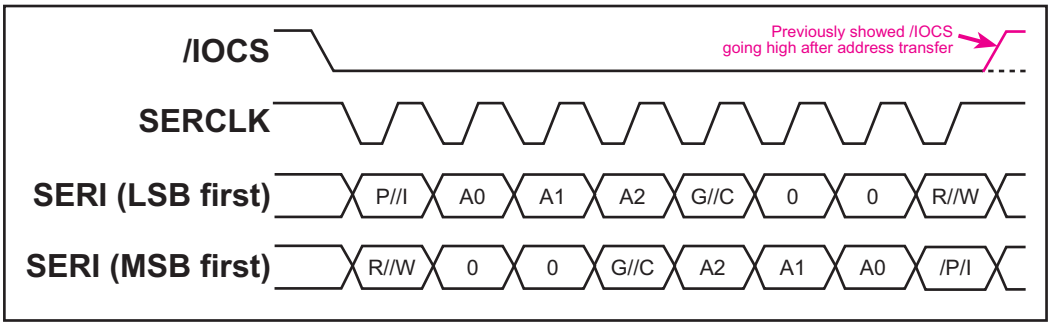
2.5.2 Serial Mode — Clocked Serial Interface

The clocked serial interface, which includes the two-wire data option (SPI) and the one-wire data option (bidirectional data), is selected by tying both the SER//PAR and G//B inputs *high*. The two-wire data option (SERIAL IN and SERIAL OUT) is selected by tying the /P/I pin *low*. The *LSB-first* serial data option is selected with BLOCK[0] *low*, and the *MSB-first* serial data option is selected with BLOCK[0] *high*. The direction of serial transfer is selected with a bit in the address byte, which must be shifted into the chip as the first phase of a data transfer cycle. The /CS signal will still function as the chip select for this communication mode.

The two-wire data option (SPI) is a half duplex interface where data from the microprocessor travel to the Rabbit RIO on one line, and data from the Rabbit RIO travel to the microprocessor on another line, but not at the same time. A synchronous clock is shared between the two devices, but only the microprocessor drives that signal.

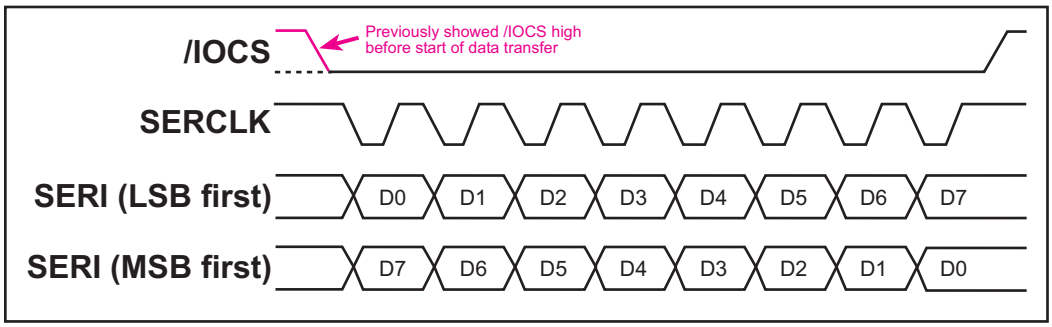
The single-wire data option works similarly to the two-wire data option, except that data are transferred between devices through one line. In this case, a transfer starts with a write operation to the Rabbit RIO to indicate whether this will be a write or a read cycle and to or from which register. In this operation, the master, typically the microprocessor, will drive the data and clock lines. If the cycle happens to be a read cycle, the address write operation will be followed by a read operation. The clock will continue to be driven by the master, but the data will be driven by the Rabbit RIO. On the other hand, if the cycle happens to be a write cycle, then the address write operation will be followed by a write operation. The clock and the data will continue to be driven by the master.

A serial mode address transfer is shown below. The five address bits function identically to the corresponding address signals in the parallel processor interface. The R//W bit controls the direction of the data transfer (high for read, low for write). Note that the value on the SERI signal is sampled by the rising edge of the SERCLK signal.

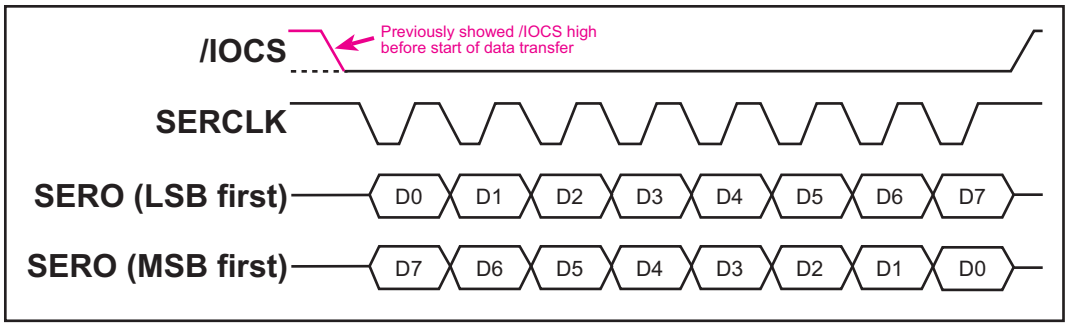


NOTE: The chip select needs to stay low between the address and data bytes of a transaction until all the data have been transferred. Therefore the chip select is shown as remaining low in the above diagram for the address transfer, then remaining low in the three diagrams that follow for the data transfer until all the data have been transferred.

A serial mode write transfer is shown below. Note that the recovery time restriction still applies in the serial bus cases, but now there is no mechanism to enforce this restriction since the /WAIT signal has become the serial output (SERO) signal. The wait time should not be an issue as long as SERCLK frequency is not more than CLK/4.



A serial mode read transfer is shown below for the case of separate serial input and output signals.



The same read transfer is shown below when using a bidirectional serial data signal. Note that any external driver on the serial data signal must be tristated before the falling edge of the serial clock. The serial data signal is driven by the device only until shortly after the rising edge of the serial clock to prevent possible bus contention.

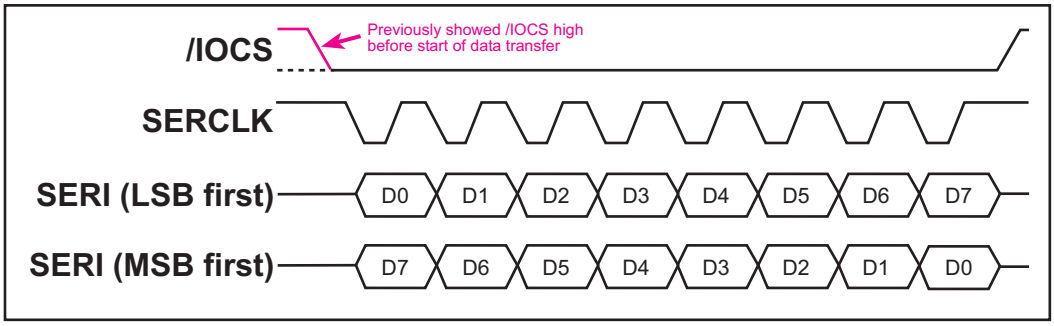


Table 2-3 lists the suggested connections to the Rabbit RIO for the SPI clocked serial interface where data flow unidirectionally on two lines (two-wire data flow).

Table 2-3. Rabbit RIO Connections — Serial Mode SPI Clocked Serial Interface

Rabbit Microprocessor Signal	Rabbit RIO Pin	Description
	SER//PAR	Pulled up
	G//B	Pulled up
	BLOCK[0]	Pulled down (LSB first) Pulled up (MSB first)
	/P/I	Pulled down
PC0, PC2, or PC4	SERI	Serial input to Rabbit RIO
PC1, PC3, or PC5	SERO	Serial output from Rabbit RIO
PD0, PD2, or PB0	SERCLK	Clock input to Rabbit RIO

Table 2-4 lists the suggested connections to the Rabbit RIO for the clocked serial interface where data flow bidirectionally on one line (one-wire data flow).

Table 2-4. Rabbit RIO Connections — Serial Mode Clocked Serial Interface (one-wire bidirectional data flow)

Rabbit Microprocessor Signal	Rabbit RIO Pin	Description
	SER//PAR	Pulled up
	G//B	Pulled up
	BLOCK[0]	Pulled down (LSB first) Pulled up (MSB first)
	/P/I	Pulled up
PC0, PC2, or PC4	SERI	Bidirectional data
PD0, PD2, or PB0	SERCLK	Clock input to Rabbit RIO

2.5.3 Serial Mode — RabbitNet Device Interface

The RabbitNet device interface option is selected by tying the SER//PAR pin *high* and the G//B and /P/I pins *low.* The table below shows the RabbitNet addressing. Additional information on RabbitNet and the RabbitNet peripheral cards is available in the ***RabbitNet Peripheral Card User's Manual***.

Table 2-5. RabbitNet Addressing on Rabbit RIO

RNA[5:0]	Selects
000000	RabbitNet Parameters (0x00)
000001	Product ID (0xF0)
000010	Reserved (0x00)
000011	Reserved (0x00)
000100	Reserved (0x00)
000101	Reserved (0x00)
000110	Reserved (0x00)
000111	Reset Status (0x00)
001xxx	Reserved (0x00)
01xxxx	Reserved (0x00)
1xxxxx	Rabbit RIO "External Address"

In order to take advantage of the SPI-like interface, the Rabbit RIO will always write back the status or echo the SERO stream, which the master can check to verify the data integrity. All data bits are transmitted and received MSB first, however, the order of bytes is little endian for RabbitNet devices transferring multiple bytes.

The protocol for communicating to the Rabbit RIO is somewhat different with RabbitNet than other forms of serial communication. To communicate with the Rabbit RIO directly from a master, a command byte must be sent out first. This byte tells the Rabbit RIO the type of transaction being initiated, determined by the two most significant bits, and which register to access, determined by the six least significant bits. At the same time the command byte is being transmitted to the Rabbit RIO, the status byte is transmitted from the Rabbit RIO. Since the status byte is transmitted automatically, its register does not have a register address. A read or a write cycle will begin immediately after the command cycle. If reading, the master will continue to drive the clock, but will be getting data driven by the Rabbit RIO. If writing, the master will drive both the clock and the data going into the Rabbit RIO.

Table 2-6 lists the suggested connections to the Rabbit RIO for the RabbitNet device interface via Serial Port C. Serial Port B or Serial Port D may also be used.

Table 2-6. Rabbit RIO Connections — RabbitNet Device Serial Interface

Rabbit Microprocessor Signal	Rabbit RIO Pin	Description
	SER//PAR	Pulled up
	G//B	Pulled down
	/P/I	Pulled down
PC2	SERI	Serial input to Rabbit RIO
PC3	SERO	Serial output from Rabbit RIO
PD2	SERCLK	Clock input to Rabbit RIO

2.5.4 Serial Mode — RabbitNet Hub Interface

The RabbitNet hub interface option is selected by tying the SER//PAR and /P/I pins *high* and tying the G//B pin *low.* The table below shows the RabbitNet hub addressing.

Table 2-7. RabbitNet Hub Addressing on Rabbit RIO

RNA[5:0]	Selects
000000	RabbitNet Parameters (0x00)
000001	Product ID (0x10)
000010	Reserved (0x00)
000011	Reserved (0x00)
000100	Reserved (0x00)
000101	Reserved (0x00)
000110	Reserved (0x00)
000111	Reset Status (0x00)
001xxx	Reserved (0x00)
01xxxx	Reserved (0x00)
1xxxxx	Rabbit RIO “External Address”

As a RabbitNet hub, the Rabbit RIO essentially becomes a different device. A hub is responsible for switching its upstream port to one of its downstream ports. The Rabbit RIO is no longer the target device, but becomes a device that reroutes signals to the appropriate RabbitNet device on its ports. A maximum of two levels of hubs are allowed in any RabbitNet network. Each hub can multiplex at most seven downstream ports.

When the master initializes the network, it must first broadcast commands to its RabbitNet port to assign each hub as a first level hub or a second level hub. It will then proceed to enumerate all devices attached to the available hubs and list them in a tabled set in memory. In this way, all devices can be easily tracked and accessed.

Table 2-8 lists the suggested connections to the Rabbit RIO for the RabbitNet hub interface via Serial Port C. Serial Port B or Serial Port D may also be used.

Table 2-8. Rabbit RIO Connections — RabbitNet Hub Serial Interface

Rabbit Microprocessor Signal	Rabbit RIO Pin	Description
	SER//PAR	Pulled up
	G//B	Pulled down
	/P/I	Pulled up
PC2	SERI	Serial input to Rabbit RIO
PC3	SERO	Serial output from Rabbit RIO
PD2	SERCLK	Clock input to Rabbit RIO

Chapter 9 provides additional information on using the Rabbit RIO as a RabbitNet hub.

2.6 Synchronization

The Rabbit RIO is non-specific to any system, meaning it will work no matter what system it is incorporated into. If the Rabbit RIO is used in a Rabbit-based system, then special logic can be used to synchronize its timing. Bit 1 of the Master Control Register can be reset to disable synchronizing logic when using a non-Rabbit-based system, or set to enable the special synchronization logic when used in a Rabbit-based system.

The synchronous bus timing option can be used when the Rabbit RIO is connected to a Rabbit microprocessor, and the RIO clock is supplied by the CLK output of the Rabbit microprocessor. The option reduces the access recovery time when the Rabbit RIO is communicating directly with a Rabbit microprocessor so that back-to-back reads and writes can be supported. Much of the synchronization logic in the Rabbit RIO can be bypassed in this case because the phase relationships for the address, data, and control signals are already known. The bit is ignored in the serial communication modes and the RabbitNet mode.

2.7 Interrupts

Interrupts can be enabled on the Rabbit RIO as an alternative to polling to provide a more efficient use of processor time. Master-level registers provide a means for interrupt control, but the exact nature of the interrupt is determined by block-level interrupt registers. To enable interrupts, Bit 0 of the Master Control Register must be set. When an I/O block triggers an interrupt, the Rabbit RIO will pull /INT low. Upon receiving the interrupt, the master can read the Master Status Register from the Rabbit RIO to determine which I/O block(s) invoked the interrupt, service the interrupt, and clear that particular interrupt in the block's Status Register.

2.8 Registers

The external address bus selects the registers according to Table 2-9 below. Each I/O block is accessed indirectly through just two external I/O addresses.

The addressing in the RabbitNet mode is similar, except the RabbitNet addresses are six bits wide, with many of the lowest address reserved for commands and configuration. In this case, the Rabbit RIO addresses are relocated to the upper 32 bytes of the RabbitNet address space, that is, the most significant bit of the RabbitNet address is set to one to access these Rabbit RIO registers (see Table 2-7).

Table 2-9. External I/O Register Addresses

Pins			Selects	R/W	Reset
G//B	BLOCK[2:0]	/P/I			
1	000	0	Master Control Register	R/W	00000000
1	000	1	Master Status Register	R/W	00000000
1	001	0	Master Prescale Register	W	00000000
1	001	1	Master Alternate Data Register	R/W	00000000
1	010	0	Master Protection Command Register	W	00000000
1	010	1	Master Protection Prescale Register	W	00000000
1	110	0	Watchdog Timer 0 Register	R/W	00000000
1	110	1	Watchdog Timer 1 Register	R/W	00000000
1	111	0	Watchdog Timer 2 Register	R/W	00000000
0	000	0	Block 0 Pointer Register	R/W	00000000
0	000	1	Block 0 Indirect Register	R/W	xxxxxxxx
0	001	0	Block 1 Pointer Register	R/W	00000000
0	001	1	Block 1 Indirect Register	R/W	xxxxxxxx
0	010	0	Block 2 Pointer Register	R/W	00000000

Table 2-9. External I/O Register Addresses (continued)

Pins			Selects	R/W	Reset
G//B	BLOCK[2:0]	/P/I			
0	010	1	Block 2 Indirect Register	R/W	xxxxxxxx
0	011	0	Block 3 Pointer Register	R/W	00000000
0	011	1	Block 3 Indirect Register	R/W	xxxxxxxx
0	100	0	Block 4 Pointer Register	R/W	00000000
0	100	1	Block 4 Indirect Register	R/W	xxxxxxxx
0	101	0	Block 5 Pointer Register	R/W	00000000
0	101	1	Block 5 Indirect Register	R/W	xxxxxxxx
0	110	0	Block 6 Pointer Register	R/W	00000000
0	110	1	Block 6 Indirect Register	R/W	xxxxxxxx
0	111	0	Block 7 Pointer Register	R/W	00000000
0	111	1	Block 7 Indirect Register	R/W	xxxxxxxx

2.9 Register Descriptions

A number of external addresses are used for registers that provide global control and status for the Rabbit RIO. Master-level registers are accessed directly through the address bus, and block-level registers are accessed indirectly through the pointer registers.

2.9.1 Master Control Register

The Master Control Register has three functions:

- provide software reset,
- set bus synchronization,
- enable or disable interrupts.

When the Rabbit RIO is interfaced with a Rabbit microprocessor, faster operation is possible by enabling synchronous bus timing via bit 1. This option can be used when the Rabbit RIO is connected to a Rabbit microprocessor, and the RIO clock is supplied by the CLK output of the Rabbit microprocessor.

Master Control Register (MCR) (External Address = 0x10)		
Bit(s)	**Value**	**Description**
7	0	No effect on the device.
	1	Reset the entire device.
6:2		These bits are ignored during writes and always return zeros when read.
1	0	Normal bus timing (used with non-Rabbit hosts).
	1	Synchronous bus timing (Rabbit 2000/3000/4000).
0	0	Disable interrupts for this device.
	1	Enable interrupts for this device.

2.9.2 Master Status Register

The Master Status Register allows a processor to determine which I/O block signaled an interrupt.

Master Status Register (MSR) (External Address = 0x11)		
Bit(s)	**Value**	**Description**
7	0	No interrupt pending for Block 7.
	1	Interrupt pending for Block 7.
6	0	No interrupt pending for Block 6.
	1	Interrupt pending for Block 6.
5	0	No interrupt pending for Block 5.
	1	Interrupt pending for Block 5.
4	0	No interrupt pending for Block 4.
	1	Interrupt pending for Block 4.
3	0	No interrupt pending for Block 3.
	1	Interrupt pending for Block 3.
2	0	No interrupt pending for Block 2.
	1	Interrupt pending for Block 2.
1	0	No interrupt pending for Block 1.
	1	Interrupt pending for Block 1.
0	0	No interrupt pending for Block 0.
	1	Interrupt pending for Block 0.

2.9.3 Master Prescale Register

Each of the eight I/O blocks has the option of either using the master clock directly or using a scaled version of that clock determined by the Master Prescale Register. The prescale value will apply universally to all the I/O blocks.

Master Prescale Register (MPR) (External Address = 0x12)		
Bit(s)	**Value**	**Description**
7:0		Time constant for the counter prescaler. This time constant will take effect the next time that the prescaler counts down to zero. The prescaler counts modulo n+1, where n is the programmed time constant. The output of the prescaler is also used to sample the input signals to detect edges.

2.9.4 Master Alternate Data Register

In the serial mode, certain pins that were used in the parallel mode for bus control are available as general-purpose input-only pins. The Master Alternate Data Register will allow the processor to read the state of these pins.

Master Alternate Data Register (MADR) (External Address = 0x13)		
Bit(s)	**Value**	**Description**
7	Read	Current state of SER//PAR pin.
6	Read	Current state of /IORD pin (serial mode only).
5	Read	Current state of /IOWR pin (serial mode only).
4	Read	Current state of G//B pin (serial mode only).
3	Read	Current state of BLOCK[2] pin (serial mode only).
2	Read	Current state of BLOCK[1] pin (serial mode only).
1	Read	Current state of BLOCK[0] pin (serial mode only).
0	Read	Current state of /P/I pin (serial mode only).

Bits 7, 4, 1, and 0 are used to establish the type of serial communication interface, and their pins are not generally used as general-purpose inputs. These pins are used to set the communication mode, and their use should not be changed since that would affect the chip’s behavior.

2.9.5 Master Protection Command Register

Pin-pair protection is enabled by writing to the Master Protection Command Register to protect the upper two bits or the lower two bits of any I/O block. The “safe” state is considered to be the state of the protected pins at the time this register is written to; therefore, an “unsafe” state is the bitwise complement of that output. The other two possible bit combinations are considered “active” states since they are allowable outputs.

Master Protection Command Register (MPCR) (External Address = 0x14)		
Bit(s)	**Value**	**Description**
7:4		These bits are reserved and should not be used.
3:0		Writing to this register enables the hardware protection for a pair of pins. This protection can only be enabled. Removing the protection requires a hardware reset. This command samples the state of the selected pins to determine the “safe” state. Hardware then enforces this “safe” state between any possible transitions on these pins when they are used as outputs.
3:1	000	Enable pin-pair protection in Block 0.
	001	Enable pin-pair protection in Block 1.
	010	Enable pin-pair protection in Block 2.
	011	Enable pin-pair protection in Block 3.
	100	Enable pin-pair protection in Block 4.
	101	Enable pin-pair protection in Block 5.
	110	Enable pin-pair protection in Block 6.
	111	Enable pin-pair protection in Block 7.
0	0	Enable protection for Pin[1:0].
	1	Enable protection for Pin[3:2].

For example, let’s say that "11" is the disallowed state (but "00", "01", and "10" are allowed). To enable pin-pair protection on two pins of an I/O block, set the output on both pins to 0 (the “safe” state, which is the complement of the disallowed state) and write the appropriate value to MPCR to enable pin-pair protection. Once this value is written, the protection will be enabled for the selected pair of pins, resulting in the following outputs.

Written to Pin	Output from Pin
00	00
01	01
10	10
11 (disallowed output)	00 (safe output)

NOTE: When pin-pair protection is enabled, the pins will be forced to the safe state between each output transition for a period determined by the Master Protection Prescale Register.

2.9.6 Master Protection Prescale Register

The value written to this register determines the dead-time time constant, a number between 0 and 255. The dead time is useful in preventing overlap during pin-output transitions. When pin-pair protection is enabled on a pair of pins, the Rabbit RIO forces the pins into a “safe” state during the dead time between signal transitions on those pins.

The main Rabbit RIO clock will be divided by (n + 1) to give the dead-time time constant. The actual dead-time will vary between (n + 1) and (2*n + 2) Rabbit RIO clocks.

Master Protection Prescale Register (MPPR) (External Address = 0x15)		
Bit(s)	**Value**	**Description**
7:0		Time constant for the protection prescaler. The prescaler counts modulo n + 1, where n is the programmed time constant. The output of the prescaler is used to time the dead time for the pin-pair protection. Since this prescaler runs continuously, the dead time is guaranteed to be at least one time constant and no more than two time constants.

For example, let’s say that "11" is the disallowed state ("00" is then the “safe” state, and "01" and "10" are allowed). When pin-pair protection is enabled for a pair of pins, to switch from "01" to "10", the output will first switch from "01" to "00" for the dead-time specified in MPPR, and then to "10".

2.9.7 Watchdog Timer Registers

All RabbitNet devices and hubs implement a watchdog timer. This timer is restarted on receipt of any command from the master. If it times out, then the device assumes that the communication link has been lost, and the device places itself in a fail-safe state. Once in the fail-safe state, the device must ignore any commands that would cause its output state to change until a soft reset is triggered through the RabbitNet Reset Status Register. This watchdog register is enable only when the Rabbit RIO is being used in the RabbitNet communication mode.

Watchdog Timer x Registers (WDT0R) (External Address = 0x1C) (WDT1R) (External Address = 0x1D) (WDT2R) (External Address = 0x1E)		
Bit(s)	**Value**	**Description**
7:0		These registers return the current count of the 23-bit watchdog timer counter, and are primarily for testing, so the entire 23-bit value is not latched. An individual byte may still be used as a pseudo-random value.

2.9.8 Pointer Registers

Each I/O block has a Pointer Register that will allow access to the block's internal registers. The lower five bits of this register holds the pointer (or address) to the internal registers. The most significant bit allows auto-incrementing of this pointer for fast configuration.

Pointer Register x	(PR0) (External Address = 0x00) (PR1) (External Address = 0x02) (PR2) (External Address = 0x04) (PR3) (External Address = 0x06) (PR4) (External Address = 0x08) (PR5) (External Address = 0x0A) (PR6) (External Address = 0x0C) (PR7) (External Address = 0x0E)	
Bit(s)	**Value**	**Description**
7	0	Disable pointer auto-increment.
	1	Enable pointer auto-increment.
6:5		These bits are ignored during writes and always return zero when read.
4:0		These five bits hold the block register address for indirect access.

2.9.9 Indirect Registers

The Indirect Register of each I/O block allows reads and/or writes to the address pointed to by the Pointer Register.

Indirect Register x	(IR0) (External Address = 0x01) (IR1) (External Address = 0x03) (IR2) (External Address = 0x05) (IR3) (External Address = 0x07) (IR4) (External Address = 0x09) (IR5) (External Address = 0x0B) (IR6) (External Address = 0x0D) (IR7) (External Address = 0x0F)	
Bit(s)	**Value**	**Description**
7:0	Read	Data from the register addressed by the Pointer Register are returned.
	Write	Data is written to the register addressed by the Pointer Register.

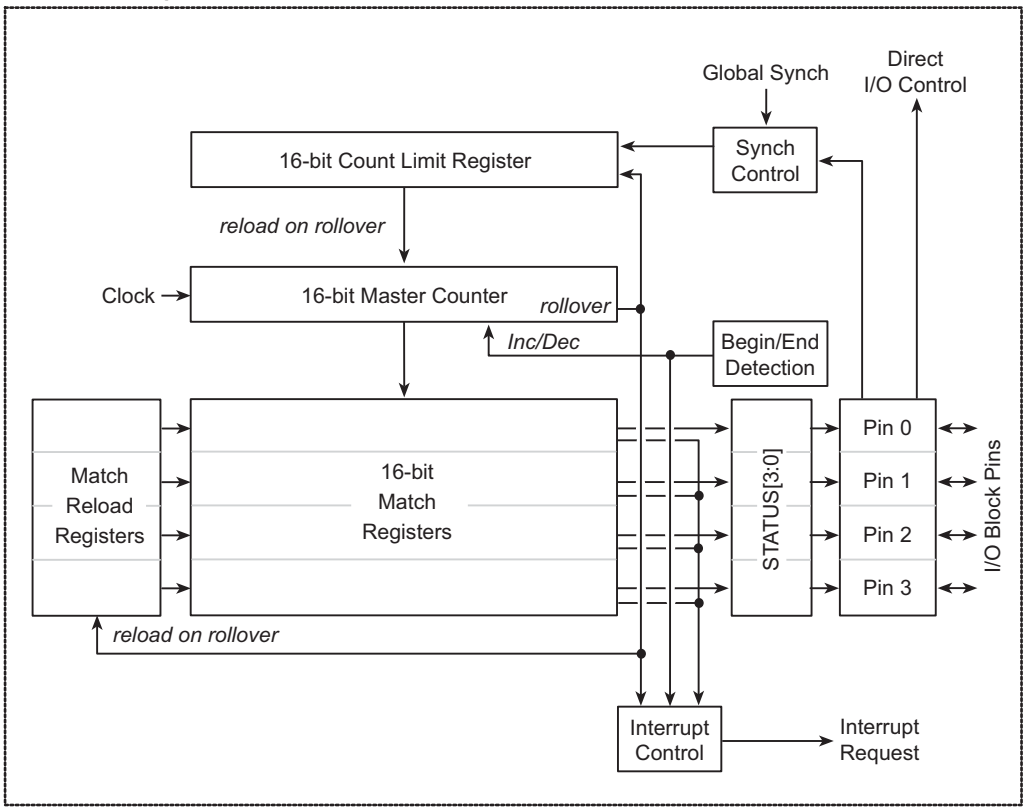
3. BLOCK-LEVEL FEATURES

3.1 Overview

The Rabbit RIO has eight identical I/O blocks. Although these I/O blocks can operate in a number of different modes, the core of each I/O block is a 16-bit counter that can be clocked by the master clock for the device, by the 8-bit Master Prescaler, or by an external source. This counter is accompanied by a number of registers that are updated with control register values each time the counter rolls over. This buffering allows the control register to be updated during the current count cycle with values to be used during the next count cycle. The registers that are buffered this way are the 16-bit reload register and the four 16-bit match registers. The four 16-bit match registers each generate an output pulse when the count is equal to the value programmed into that register. These pulses can be used to set or reset any of the outputs from the I/O block. The 16-bit limit register determines when the counter will be reloaded. Some of the different operating modes that are possible are described in more detail in the later chapters.

3.1.1 Simplified Block Diagram

Rabbit RIO Chip Block-Level Features



3.2 Internal Block Registers

Register Name	Mnemonic	Address	R/W	Reset
Pointer Register	PR	External	R/W	00000000
Indirect Register	IR	External	R/W	xxxxxxxx
Command Register	CR	PR=0x00	W	11111111
Mode Register	MR	PR=0x01	R/W	00000000
Interrupt Enable Register	IER	PR=0x02	R/W	00000000
Status Register	SR	PR=0x03	R/W	00000000
Counter Toggle Register	CTR	PR=0x04	W	00000000
Synch Control Register	SCR	PR=0x05	W	00000000
Increment/In-Phase/Begin Control Register	ICR	PR=0x06	W	00000000
Decrement/Quadrature/End Control Register	DCR	PR=0x07	W	00000000
Status 0 Control Register	S0CR	PR=0x08	W	00000000
Status 1 Control Register	S1CR	PR=0x09	W	00000000
Status 2 Control Register	S2CR	PR=0x0A	W	00000000
Status 3 Control Register	S3CR	PR=0x0B	W	00000000
Pin 0 Control Register	P0CR	PR=0x0C	R/W	00000000
Pin 1 Control Register	P1CR	PR=0x0D	R/W	00000000
Pin 2 Control Register	P2CR	PR=0x0E	R/W	00000000
Pin 3 Control Register	P3CR	PR=0x0F	R/W	00000000
Match 0 LSB Register	M0LR	PR=0x10	W	00000000
Match 0 MSB Register	M0MR	PR=0x11	W	00000000
Match 1 LSB Register	M1LR	PR=0x12	W	00000000
Match 1 MSB Register	M1MR	PR=0x13	W	00000000
Match 2 LSB Register	M2LR	PR=0x14	W	00000000
Match 2 MSB Register	M2MR	PR=0x15	W	00000000
Match 3 LSB Register	M3LR	PR=0x16	W	00000000
Match 3 MSB Register	M3MR	PR=0x17	W	00000000
Count Limit LSB Register	CLLR	PR=0x18	W	00000000
Count Limit MSB Register	CLMR	PR=0x19	W	00000000
Count Begin LSB Register	CBLR	PR=0x1A	R	00000000
Count Begin MSB Register	CBMR	PR=0x1B	R	00000000
Count End LSB Register	CELR	PR=0x1C	R	00000000
Count End MSB Register	CEMR	PR=0x1D	R	00000000
Count Value LSB Register	CVLR	PR=0x1E	R	00000000
Count Value MSB Register	CVMR	PR=0x1F	R	00000000

3.3 Block Control

Each I/O block is controlled through a set of registers, listed in Section 3.2. Only the Pointer Register and the Indirect Register are accessible directly from the external interface. All the other registers within the I/O block are accessed by first writing the register address to the Pointer Register and then reading from or writing to the Indirect Register. Reading from or writing to the Indirect Register can increment the Pointer Register automatically to address the next sequential register in the I/O block if bit 7 of the Pointer Register is set. This feature can reduce the number of address writes required when accessing multi-byte values.

3.4 Register Descriptions

3.4.1 Pointer and Indirect Registers

Each I/O block can only be accessed through the block's Pointer Register and Indirect Register. The Pointer Registers specify the addresses of the internal registers, and the Indirect Registers access the data for that internal register. When the most significant bit of the Pointer Register is set by writing the internal register address, the Pointer Register will increment automatically after each read or write to the Indirect Register.

Pointer Register x		**(PR0) (External Address = 0x00)** **(PR1) (External Address = 0x02)** **(PR2) (External Address = 0x04)** **(PR3) (External Address = 0x06)** **(PR4) (External Address = 0x08)** **(PR5) (External Address = 0x0A)** **(PR6) (External Address = 0x0C)** **(PR7) (External Address = 0x0E)**
Bit(s)	**Value**	**Description**
7	0	Disable pointer auto-increment.
	1	Enable pointer auto-increment.
6:5		These bits are ignored during writes and always return zero when read.
4:0		These five bits hold the block register address for indirect access.

Indirect Register x		**(IR0) (External Address = 0x01)** **(IR1) (External Address = 0x03)** **(IR2) (External Address = 0x05)** **(IR3) (External Address = 0x07)** **(IR4) (External Address = 0x09)** **(IR5) (External Address = 0x0B)** **(IR6) (External Address = 0x0D)** **(IR7) (External Address = 0x0F)**
Bit(s)	**Value**	**Description**
7:0	Read	Data from the register addressed by the Pointer Register are returned.
	Write	Data is written to the register addressed by the Pointer Register.

3.4.2 Command Register

The Command Register can internally generate a signal that would otherwise come from an external source. For example, it can force a synch signal that would normally come from the GSYNC pin or from any of the I/O block's pins.

Command Register (CR) (Pointer = 0x00)		
Bit(s)	**Value**	**Description**
7	0	No operation. This bit always reads a zero.
	1	Reset counter to all zeros.
6	0	No operation. This bit always reads a zero.
	1	Force a synch signal.
5	0	No operation. This bit always reads a zero.
	1	Force an Increment/Begin signal.
4	0	No operation. This bit always reads a zero.
	1	Force a Decrement/End signal.
3:0		These bits are reserved an should not be used.

3.4.3 Mode Register

The Mode Register defines the overall function of the I/O block by specifying how the 16-bit counter will be used. All counter modes increment or decrement the counter from external signal events brought in on one of the four I/O block pins. All timer modes increment the counter from either the master or the prescaled RIO clock

Mode Register (MR) (Pointer = 0x01)		
Bit(s)	**Value**	**Description**
7:4		These bits are ignored and should not be used.
3	0	Use the master clock directly (for both input synchronizers and timer).
	1	Use prescaler output clock (for both input synchronizers and timer).
2:0	000	Disable counter/timer.
	001	Enable counter mode. Increment count forever on each Increment/ In-Phase/Begin signal.
	010	Enable counter mode. Increment count until any match condition on each Increment/In-Phase/Begin signal.
	011	Enable counter mode. Count up on each Increment/In-Phase/Begin and count down on each Decrement/Quadrature/End.
	100	Enable timer mode. Counter increments continuously.
	101	Enable timer mode. Counter increments continuously, until the End condition.
	110	Enable timer mode. Counter increments from Begin condition until the End condition.
	111	Enable timer mode. Counter increments continuously during the Begin condition (uses begin level, not edge detection).

3.4.4 Interrupt Enable and Status Registers

Controlling I/O block interrupts is a matter of writing to the proper bits of the Interrupt Enable Register (IER). The generation of an interrupt defined by the IER is registered to the block's Status Register (SR). The block interrupt will in turn be registered to the Master Status Register (MSR).

Interrupt Enable Register (IER) (Pointer = 0x02)		
Bit(s)	**Value**	**Description**
7	0	Disable Decrement/Quadrature/End interrupt.
	1	Enable Decrement/Quadrature/End interrupt
6	0	Disable Increment/In-Phase/Begin interrupt.
	1	Enable Increment/In-Phase/Begin interrupt.
5	0	Disable counter rollover (decrement) interrupt.
	1	Enable counter rollover (decrement) interrupt.
4	0	Disable counter rollover (increment) interrupt.
	1	Enable counter rollover (increment) interrupt.
3	0	Disable Match 3 interrupt.
	1	Enable Match 3 interrupt.
2	0	Disable Match 2 interrupt.
	1	Enable Match 2 interrupt.
1	0	Disable Match 1 interrupt.
	1	Enable Match 1 interrupt.
0	0	Disable Match 0 interrupt.
	1	Enable Match 0 interrupt.

Status Register (SR) (Pointer = 0x03)		
Bit(s)	**Value**	**Description**
7:0 Write	0	No operation
	1	Clear the corresponding interrupt condition
7 Read	0	No Decrement/Quadrature/End interrupt condition pending.
	1	Decrement/Quadrature/End interrupt condition pending.
6 Read	0	No Increment/In-Phase/Begin interrupt condition pending
	1	Increment/In-Phase/Begin interrupt condition pending.
5 Read	0	No counter rollover (increment) interrupt condition pending.
	1	Counter rollover (increment) interrupt condition pending.
4 Read	0	No Counter Rollover (decrement) interrupt condition pending.
	1	Counter Rollover (decrement) interrupt condition pending.
3 Read	0	No Match 3 interrupt condition pending.
	1	Match 3 interrupt condition pending.
2 Read	0	No Match 2 interrupt condition pending.
	1	Match 2 interrupt condition pending.
1 Read	0	No Match 1 interrupt condition pending.
	1	Match 1 interrupt condition pending.
0 Read	0	No Match 0 interrupt condition pending.
	1	Match 0 interrupt condition pending.

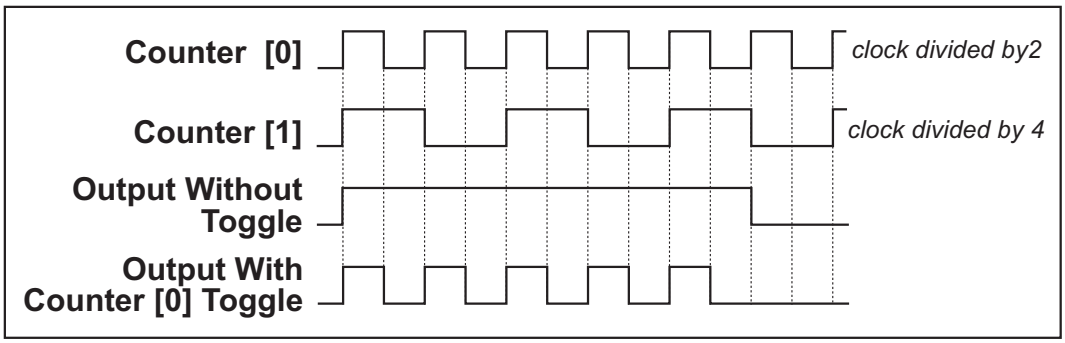
3.4.5 Counter Toggle Register

The toggle function imposes a fixed frequency/duty cycle “mask” over the output when it would otherwise be continuously high. The duty cycle may be set to 50%, 25%, 12.5%, and so on by writing to this register. The purpose of this function is generally to reduce driver current for applications such as relay coils or thyristor gate drivers that control loads.

Counter Toggle Register	(CTR)	(Pointer = 0x04)
Bit(s)	**Value**	**Description**
7:0	0	Deselect the corresponding bit of the counter as the toggle signal.
	1	Select the corresponding bit of the counter as the toggle signal. Select one bit for a 50% duty cycle. Select two adjacent bits for a 25% duty cycle, and so on. The actual toggle signal is generated as Toggle = &(~CTR | Counter[7:0]).

The output will toggle whenever the pin is supposed to output a logic one.

The diagram below shows how Counter [0] and Counter [1] would look relative to each other, and then shows the effects of toggling an output with Counter [0].



3.4.6 Synch Control Register

The Synch Control Register is used to select the pin that will be used as the synch signal for that particular I/O block. A pin used as a synch signal can reset the block's counter. Similarly, the GSYNC signal has the ability to reset all the I/O blocks on the Rabbit RIO if it is selected to do so.

Synch Control Register (SCR) (Pointer = 0x05)		
Bit(s)	**Value**	**Description**
7	0	Synch signal does not affect counter.
	1	Enable synch signal to reset counter.
6		This bit is reserved and should not be used.
5:3	000	Disable synch signal, without generating an edge.
	001	Synch signalled on rising edge.
	010	Synch signalled on falling edge.
	011	Synch signalled on either edge.
	100	This bit combination is reserved and should not be used.
	101	This bit combination is reserved and should not be used.
	110	Synch signalled while low.
	111	Synch signalled while high.
2:0	000	No selection. Selector output is low.
	001	This bit combination is reserved and should not be used.
	010	This bit combination is reserved and should not be used.
	011	Select Global Synch.
	100	Select Pin[0].
	101	Select Pin[1].
	110	Select Pin[2].
	111	Select Pin[3].

3.4.7 Increment/In-Phase/Begin Control Register

This register controls which pin is used to increment the counter, act as the in-phase input to a quadrature decoder, or begin the counter. It also controls how that pin is detected, whether it be by an edge, by a level, or by a transition.

Increment/In-Phase/Begin Control Register (ICR) (Pointer = 0x06)		
Bit(s)	**Value**	**Description**
7:6		These bits are reserved and should not be used.
5:3	000	Disable Increment/In-Phase/Begin signal without generating an edge
	001	Increment/In-Phase/Begin signal on rising edge.
	010	Increment/In-Phase/Begin signal on falling edge.
	011	Increment/In-Phase/Begin signal on either edge.
	100	Increment/In-Phase/Begin signal on 10 → 11 → 01 → 00 → 10 quadrature decoder transitions.
	101	This bit combination is reserved and should not be used.
	110	Increment/In-Phase/Begin signal while low.
	111	Increment/In-Phase/Begin signal while high.
2:0	000	No selection. Selector output is low.
	001	This bit combination is reserved and should not be used.
	010	This bit combination is reserved and should not be used.
	011	Select Global Synch.
	100	Select Pin[0].
	101	Select Pin[1].
	110	Select Pin[2].
	111	Select Pin[3].

3.4.8 Decrement/Quadrature/End Control Register

Similar to the Increment/In-Phase/Begin Control Register (ICR), the Decrement/Quadrature/End Control Register (DCR) complements the ICR in its function to decrement the counter, act as the quadrature input to a quadrature signal, or stop the counter. It also controls how that pin is detected, whether it be by an edge, level, or transition.

Decrement/Quadrature/End Control Register (DCR) (Address = 0x07)		
Bit(s)	**Value**	**Description**
7:6		These bits are reserved and should not be used.
5:3	000	Disable Decrement/Quadrature/End signal without generating an edge
	001	Decrement/Quadrature/End signalled on rising edge.
	010	Decrement/Quadrature/End signalled on falling edge.
	011	Decrement/Quadrature/End signalled on either edge.
	100	Decrement/Quadrature/End signalled on 01 → 11 → 10 → 00 → 01 quadrature decoder transitions.
	101	This bit combination is reserved and should not be used.
	110	Decrement/Quadrature/End signalled while low.
	111	Decrement/Quadrature/End signalled while high.
2:0	000	No selection. Selector output is low.
	001	This bit combination is reserved and should not be used.
	010	This bit combination is reserved and should not be used.
	011	Select Global Synch.
	100	Select Pin[0].
	101	Select Pin[1].
	110	Select Pin[2].
	111	Select Pin[3].

3.4.9 Status Control Registers

The Status Control Registers determine how the status signal is generated, if at all.

Status x Control Register		(S0CR) (Address = 0x08) (S1CR) (Address = 0x09) (S2CR) (Address = 0x0A) (S3CR) (Address =0x0B)
Bit(s)	**Value**	**Description**
7:6		These bits are reserved and should not be used.
5:3	000	Status x not set by internal conditions.
	001	Status x set by counter rollover (decrement).
	010	Status x set by counter rollover (increment).
	011	Status x set by synch signal.
	100	Status x set by Match 0 condition.
	101	Status x set by Match 1 condition.
	110	Status x set by Match 2 condition.
	111	Status x set by Match 3 condition.
2:0	000	Status x reset by internal conditions.
	001	Status x reset by counter rollover (decrement).
	010	Status x reset by counter rollover (increment).
	011	Status x reset by synch signal.
	100	Status x reset by Match 0 condition.
	101	Status x reset by Match 1 condition.
	110	Status x reset by Match 2 condition.
	111	Status x reset by Match 3 condition.

3.4.10 Pin Control Registers

The Pin Control Registers determine the overall function of a pin.

Pin x Control Register		(P0CR) (Address = 0x0C) (P1CR) (Address = 0x0D) (P2CR) (Address = 0x0E) (P3CR) (Address = 0x0F)
Bit(s)	**Value**	**Description**
7:4	read	These bits return the current state of Pin[3:0].
7:4	write	These bits store the output data for Pin[3:0] (P3CR only).
3		This bit is reserved and should not be used.
2:0	000	Pin bit is an input.
	001	This bit combination is reserved and should not be used.
	010	This bit is an output — Status x & Toggle.
	011	Pin bit is an output — Status x
	100	Pin bit is an output: use appropriate bit of PxCR. Sequence through P0CR-P1CR-P2CR-P3CR, changing on each counter (increment) rollover.
	101	Pin bit is an output: use appropriate bit of P3CR.
	110	Pin bit is an output — low.
	111	Pin bit is an output — high.

3.4.11 Match Registers

Match x LSB Register		(M0LR) (Address = 0x10) (M1LR) (Address = 0x12) (M2LR) (Address = 0x14) (M3LR) (Address = 0x16)
Match x MSB Register		**(M0MR) (Address = 0x11) (M1MR) (Address = 0x13) (M2MR) (Address = 0x15) (M3MR) (Address = 0x17)**
Bit(s)	**Value**	**Description**
7:0		When the current count equals the value held in this register, the Match x signal is true. This signal can be used to set a status signal and generate an interrupt request. The value in this register is loaded into a holding register for the comparison each time the counter is reloaded to allow modification of the match value during the current count cycle.

3.4.12 Count Limit Registers

The Count Limit Register is a 16-bit register used to tell the counter that the counter has reached the value held in these registers and to reset.

Count Limit LSB Register		(CLLR)	(Address = 0x18)
Count Limit MSB Register		(CLMR)	(Address = 0x19)
Bit(s)	**Value**	**Description**	
7:0		When the counter increments beyond the value held in the register pair, the counter will be reset to all zeros, effectively causing division by $n + 1$. For example, this means that placing an 8 in the register results in a divide by 9.	

3.4.13 Count Begin Registers

At the trigger of the Begin signal, the value of the counter is latched into the Count Begin Registers.

Count Begin LSB Register		(CBLR)	(Address=0x1A)
Count Begin MSB Register		(CBMR)	(Address=0x1B)
Bit(s)	**Value**	**Description**	
7:0		These registers return the count that was latched by the Begin condition. The value that is ready is guaranteed to be stable by freeing the read data while the pointer is pointing to either of these registers. This latching function is independent of the latches controlled by the Begin condition.	

3.4.14 Count End Registers

At the trigger of the End signal, the value of the counter is latched to the Count End Registers.

Count End LSB Register		(CELR)	(Address=0x1C)
Count End MSB Register		(CEMR)	(Address=0x1D)
Bit(s)	**Value**	**Description**	
7:0		These registers hold the count that was latched by the End condition. The value that is read is guaranteed to be stable by freezing the read data while the pointer is pointing to either of these registers. This latching functions is independent of the latches controlled by the End condition.	

3.4.15 Count Value Registers

At any point during the count cycle, the value of the counter can be read from this register.

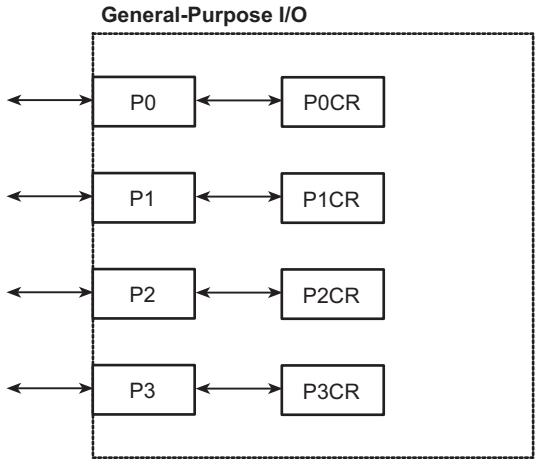
Count Value LSB Register		(CVLR)	(Address = 0x1E)
Count Value MSB Register		(CVMR)	(Address=0x1F)
Bit(s)	**Value**	**Description**	
7:0		The current count is returned. The value that is read is guaranteed to be stable by freezing the read data while the pointer is pointing to either of these registers. This implies that the pointer must be modified to point at another register to allow a new current count to be read.	

4. GENERAL-PURPOSE I/O

4.1 Overview

Each I/O block may be used as a simple *parallel input pin* or as a *parallel output pin*. The timer can still be used to generate periodic interrupts separately. The data register returns the state of the pins associated with this I/O block independent of the data direction.

4.1.1 Block Diagram



4.2 Dependencies

Item	Description	Register
Pin	I/O control	PxCR

4.3 Operation

Each I/O block has four pins that may be used for general-purpose I/O. The corresponding PxCR register can be used to access each pin separately by setting each pin to be either an input or an output. When a pin is set as an input, the state of that pin can be read in through any one of the four pin control registers. There are two ways to drive an output. The most direct way is to write a nibble to the upper four bits of P3CR, where bit 7 corresponds to pin 3 of that I/O block, bit 6 corresponds to pin 2, and so on. The alternate way is to access each of the four Pin Control Registers (P0CR, P1CR, P2CR, P3CR). To output a logic 1 on pin 3, for example, P3CR[2:0] must be written with 111. To output a logic 0 on that same pin, 110 must be written.

4.3.1 Setup

The following steps explain how to set up a general-purpose I/O pin.

1. Define the pin to be an input or an output through its corresponding Pin Control Register. Each I/O block has four pin control registers, and the Pointer Register points to the block. For example, to define Block 0, Pin 0 (B0P0), write 0x0C (P0CR) to 0x00 (PR0), and write bits [2:0] in P0CR to the corresponding Indirect Register (IR0 in this example).
2. If any of the pins are inputs, the logic value on the pins can be read in from the upper four bits of that I/O block's pin control register.
3. A single bit value can be written to a pin by writing the appropriate value to the lower three bits of its corresponding pin control register.
4. If all four pins are outputs, writing to the upper nibble of P3CR will determine the logic value on those pins only when bits [2:0] in the Pin x Control Register are 101.

4.3.2 Example of Operation

The state of each pin can be controlled individually through its corresponding Pin Control Register. To output low, the three least significant bits of the register are written to with 110b. To output high, the three least significant bits of the register are written to with 111b.

4.3.3 Pattern Mode

There is an option for a sequence of four output settings to be enabled to cycle through the bit settings in the upper nibble of the pin control registers.

4.4 Register Descriptions

Pin x Control Register		(P0CR) (Pointer = 0x0C) (P1CR) (Pointer = 0x0D) (P2CR) (Pointer = 0x0E) (P3CR) (Pointer = 0x0F)
Bit(s)	**Value**	**Description**
7:4	read	These bits return the current state of Pin[3:0].
7:4	write	These bits store the output data for Pin[3:0] (P3CR only).
3		This bit is reserved and should not be used.
2:0	000	Pin bit is an input.
	001	This bit combination is reserved and should not be used.
	010	Pin bit is an output — Status x and Toggle.
	011	Pin bit is an output — Status x.
	100	Pin bit is an output — use the appropriate bit of PxCR. Sequence through P0CR-P1CR-P2CR-P3CR, changing on each counter (increment) rollover.
	101	Pin bit is an output — use the appropriate bit of P3CR.
	110	Pin bit is an output — low.
	111	Pin bit is an output — high.

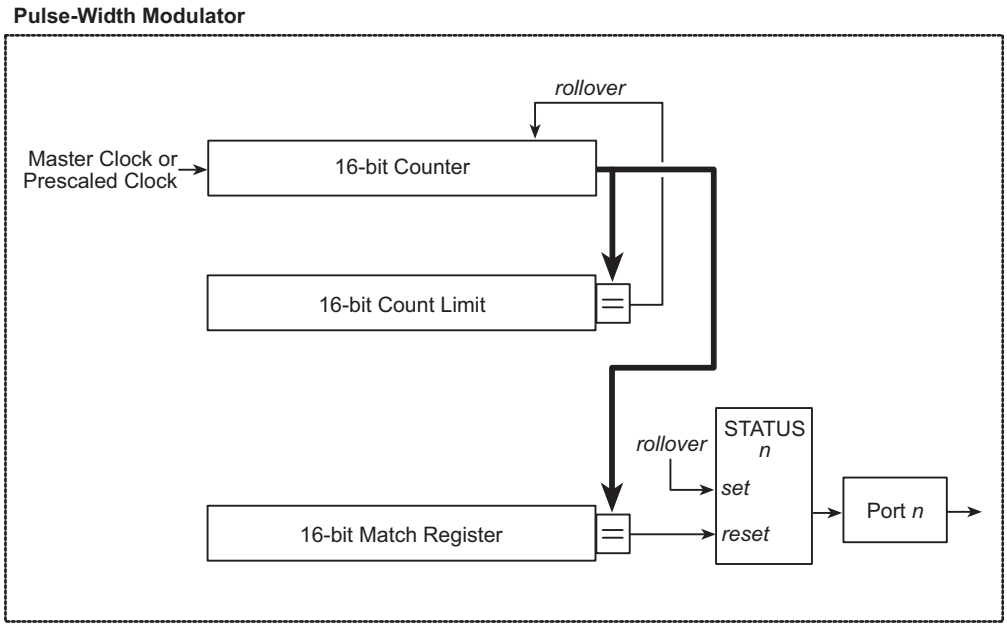
5. PULSE-WIDTH MODULATOR

5.1 Overview

The Rabbit RIO can be used to generate pulse-width modulator (PWM) signals. Each I/O block can generate up to four synchronized PWM signals that rise on the same counter rollover but have different duty cycles. Chapter 6 describes the generation of variable-phase PWM signals.

The block counter is put into the timer mode and is used to generate the period of the PWM waveform by setting both the clock input (either the master clock or the prescaler output) and the Count Limit Register. The PWM outputs are created via the status bits, which are set during the counter rollover and are reset by the Match Registers.

5.1.1 Block Diagram



5.2 Dependencies

Item	Description	Register
Counter	Yes	MPR, MR
Count Limit	Yes -- defines PWM period	CLLR, CLMR
Match Register	One per PWM signal (up to 4)	MnLR, MnMR
Status Bit	One per PWM signal (up to 4)	SnCR
Pin	One per PWM output (up to 4)	PnCR
Synch	Optional — used to reset counter	SCR
Interrupt	Optional — used for interrupt when match or rollover occurs	IER

5.3 Operation

When the counter is incremented by the main clock, the PWM period can be calculated as follows.

$$\text{period} = \frac{(\text{count limit} + 1)}{\text{main clock frequency}}$$

When the counter is incremented by the prescaled clock, the following equation should be used.

$$\text{period} = \frac{(\text{count limit} + 1) \times (\text{prescaler value} + 1)}{\text{main clock frequency}}$$

5.3.1 Setup

The following steps explain how to set up an I/O block for PWM operation.

1. Select the clock by writing to the Master Prescale Registers and to the Block Mode Register.
2. Set the Count Limit Registers to determine the PWM period.
3. Set the Match Registers to the desired duty cycles.
4. Set the status bit and pin for the desired outputs by writing to the Status n Control Register and to the Pin n Control Register.

5.3.2 Example

Figure 5-1 shows a sample PWM output with the registers written to as follows.

Register	Value	Description
MR	0x0C	Use prescaled clock, enable timer mode, continuous count
MPR	0x13	Divide 20 MHz clock by 20 (MPR = 19).
CLLR CLMR	0x270F	Count limit is 10,000 (CLR = 9,999).
M0LR M0MR	0x1388	Match at 5,000 to give 50-50 duty cycle.
S0CR	0x22	Status 0 set by Match 0, reset by counter rollover (increment).
P0CR	0x03	Pin bit outputs status.

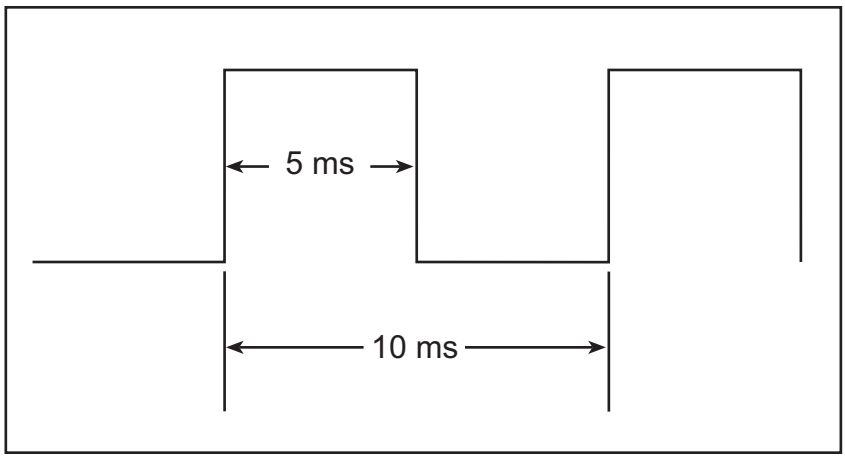


Figure 5-1. Sample PWM Output

5.4 Other Comments

5.4.1 General-Purpose I/O

Any pins not in use for PWM can be used for general-purpose I/O.

5.4.2 External Synchronization

The PWM signals can be synchronized to an external signal via either the global synch or a local I/O block pin by writing to the Synch Control Register. This feature can be used to generate triac control signals by sending a zero-crossing signal to the Global Synch pin.

5.4.3 Interrupts

If desired, an interrupt can be generated from either the counter rollover or whenever any of the match registers match.

5.4.4 Higher Drive Current Operations

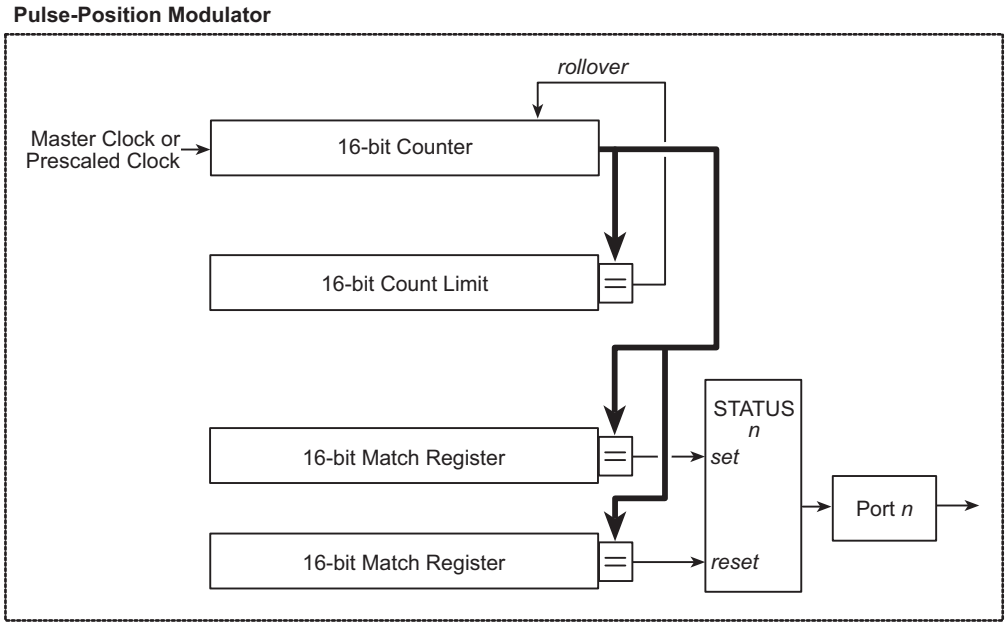
Multiple status bits could be set from a single match register, so the same signal could be output on multiple pins for applications that require a higher drive current.

6. VARIABLE-PHASE PULSE-WIDTH MODULATOR

6.1 Overview

A variable-phase PWM, or Pulse-Position Modulator (PPM), is simply a PWM that can shift in phase. Unlike a PWM whose phase is referenced to the counter rollover, the PPM can be triggered high and low by a combination of the match registers. Just like the PWM, the block counter is put into the timer mode and is used to generate the period of the waveform by setting both the clock input and the Count Limit Register. The PPM outputs are created via the status bits, which are set and reset by any of the four match registers in the I/O block.

6.1.1 Block Diagram



6.2 Dependencies

Item	Description	Register
Counter	Yes	MPR, MR
Count Limit	Yes — defines PPM period	CLLR, CLMR
Match Register	Two per PPM signal (up to 2 per I/O block)	MnLR, MnMR
Status Bit	Two per PPM signal (up to 2 per I/O block)	SnCR
Pin	One per PPM output	PnCR
Synch	Optional — used to reset counter	SCR
Interrupt	Optional — used for interrupt when match or rollover occurs	IER

6.3 Operation

When the counter is incremented by the main clock, the PPM period can be calculated as follows.

$$\text{period} = \frac{(\text{count limit} + 1)}{\text{main clock frequency}}$$

When the counter is incremented by the prescaled clock, the following equation should be used.

$$\text{period} = \frac{(\text{count limit} + 1) \times (\text{prescaler value} + 1)}{\text{main clock frequency}}$$

The above equation translates to the actual time of the period, not the “count limit” value used to determine the period; however, the equation can be rearranged to give the “count limit” value. The calculation below shows where to start and stop for a desired duty cycle.

$$\text{start PPM} = 0$$

$$\text{end PPM} = \frac{(\text{period} \times \text{duty cycle})}{100}$$

where

“duty cycle” is amount of time the signal is on, expressed as a percentage (40% = 40), and

“period” is the counter value, not actually time, that is placed into the Count Limit Register.

To change the phase of the calculated start and end counts,

$$\text{start} = \left(\frac{\text{start PPM} + \text{phase} \times \text{period}}{360}\right) \bmod \text{period}$$

where 0 < phase < 360° and

$$\text{end} = \left(\frac{\text{end PPM} + \text{phase} \times \text{period}}{360}\right) \bmod \text{period}$$

where 0 < phase < 360°.

The “start” and “end” values are placed into any of the Match Registers.

6.3.1 Setup

The following steps explain how to set up an I/O block for PPM operation.

1. Select the clock by writing to the Master Prescale Registers and to the Block Mode Register.
2. Set the Count Limit Registers to determine the PPM period.
3. Set the Match Registers to the desired duty cycles and phase as calculated above.
4. Set the status bit and the pin for the desired outputs by writing to the Status n Control Register and to the Pin n Control Register.

6.3.2 Example of Operation

Figure 6-1 shows a sample PPM output with the registers written to as follows.

Register	Value	Description
MR	0x0C	Use prescaled clock, enable timer mode, continuous count
MPR	0x13	Divide 20 MHz clock by 20 to give 1 µs clock.
CLLR CLMR	0x0063	Count limit is 99, which is the period of 100 µs
M0LR M0MR	0x19	Match 0 is set to 25.
M1LR M1MR	0x32	Match 1 is set to 50.
S0CR	0x25	Status set by Match 0 and reset by Match 1.
P0CR	0x03	Pin bit outputs status.

For a 25% duty cycle, a period of 100 µs, and a master clock of 20 MHz,

start = 0

end = (100 × 25) / 100 = 25

For a phase shift of 90°,

start = (0 + 90 × 100 / 360) mod 100 = 25

end = (25 + 90 × 100/ 360) mod 100 = 50

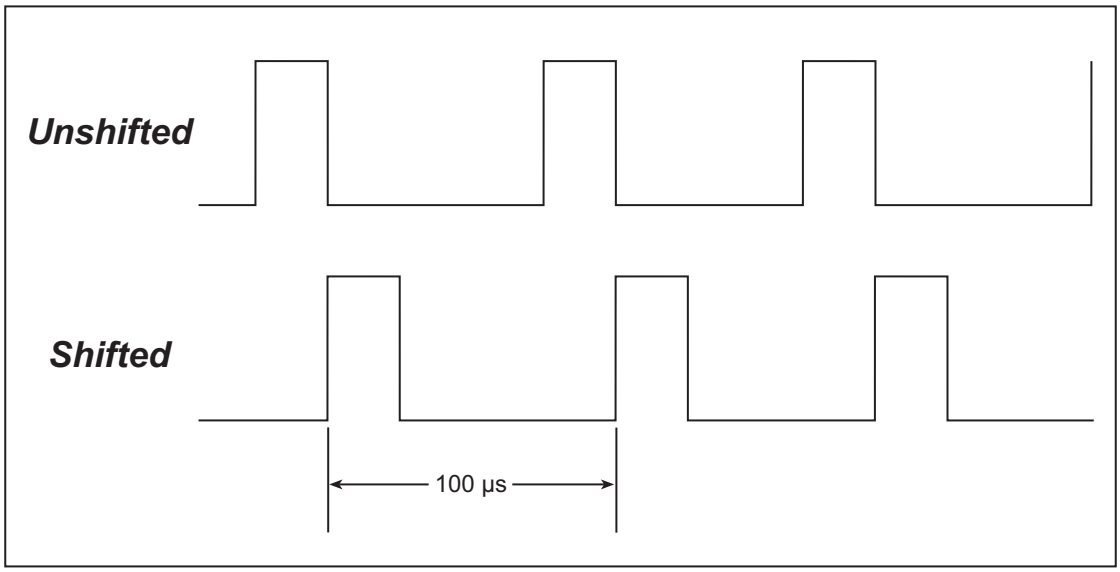


Figure 6-1. Sample PPM Output (100 µs period, 25% duty cycle)

6.4 Other Comments

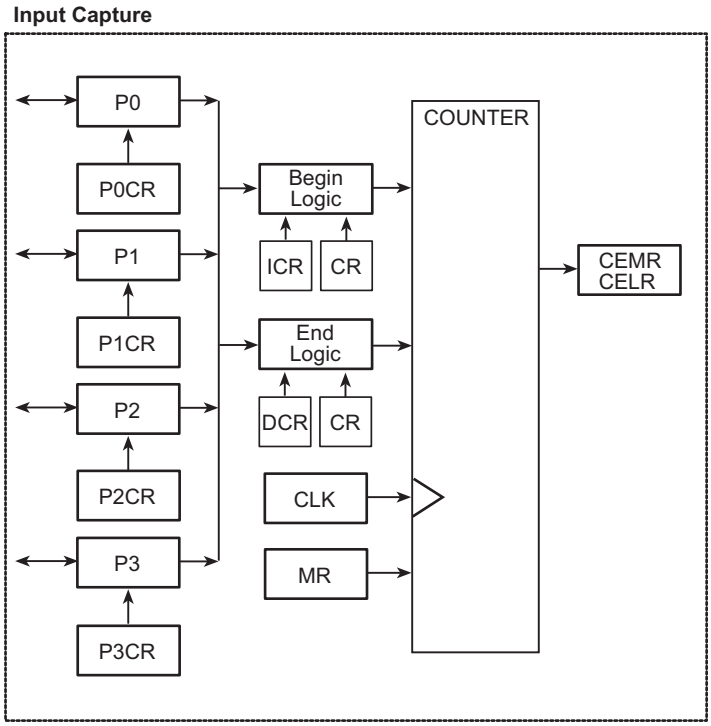
The PPM works just like the PWM, except with a little more control as to when the signal is high and low. Unlike the PWM, each I/O block can only support two of these functions, not four.

7. INPUT CAPTURE

7.1 Overview

The input capture function is useful for determining the relative timing between pulses or edges, or for determining the timing of an external signal relative to a software-generated signal. This function uses the block counter to start counting when a Begin signal is detected. It will stop when an End signal is detected.

7.1.1 Block Diagram



7.2 Dependencies

Item	Description	Register
Pin	Signals to measure	PnCR
Begin/End	Start and stop control	ICR, DCR, CR
Counter	Counter control	MPR, MR, CLKI
Count Value	Hold measured value	CEMR, CELR
Interrupt	Enable interrupt (optional)	IER

7.3 Operation

To measure the width of a pulse while the signal is high, the Rabbit RIO will count continuously while this signal is a logic 1. The counter can be clocked by either the master clock or the prescaled clock, which is determined by the Master Prescale Register. Choosing between the two clocking modes will require bit 3 of the I/O block's Mode Register to be reset or set, respectively.

To measure the time between two pulses, the Rabbit RIO will detect an edge of one signal from a pin and start counting. It will stop once it detects an edge from another specified pin.

The block counter can be reset for the next round of measurements either at the Begin signal or the End signal. The value of the counter can be read from the Count-End LSB (CELR) Register and the Count-End MSB (CEMR) Register.

An interrupt can be generated by a begin signal or by an end signal to provide an indication to the host microprocessor that a measurement is available or has started.

7.3.1 Setup

The following steps explain how to set up an input capture block.

1. Select clock by writing to the Master Prescale Registers and to the Block Mode Register.
2. Set the Count Limit Registers to a high enough value so that it does not roll over during normal operation.
3. Enable the timer mode in the Mode Register.
4. Set the begin and end signals in the Increment/In-Phase/Begin Control Register (ICR) and in the Decrement/Quadrature/End Control Register (DCR).
5. Set the pins as inputs in the Pin n Control Registers.
6. Enable the interrupt in the Interrupt Enable Register (IER) if desired.

7.3.2 Example

Figure 7-1 shows a sample input capture output with the registers written to as follows.

Register	Value	Description
MR	0x0E	Use master prescaler, count from beginning condition to end condition
MPR	0x13	Divide 20 MHz clock by 20 to get period of 1 µs.
ICR	0x0C	Begin on rising edge of Pin 0.
DCR	0x14	End on falling edge of Pin 0.
P0CR	0x00	Make Pin 0 an input.

When the signal enters the input capture pin, the resulting value read in will be 12. The counter will continue to count as long as the signal on Pin 0 is high. The value of the count can be read in from CELR and CEMR, the Count-End Registers.

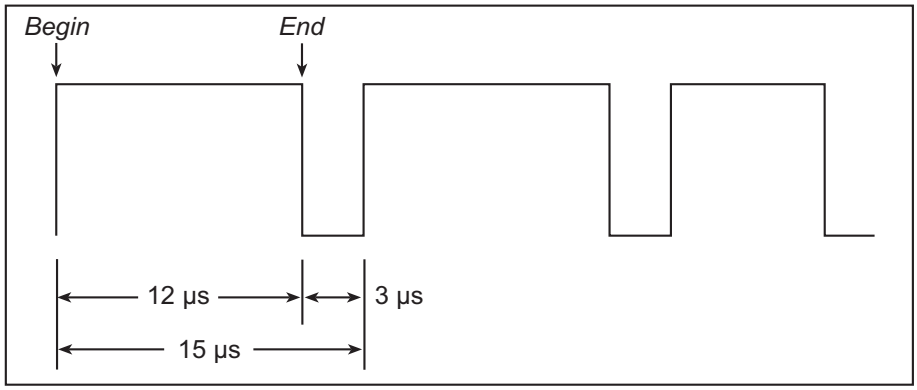


Figure 7-1. Sample Input Capture Output

NOTE: If a stop event occurs before a start event, the counter will stop. You must synchronize arming the event counters so that the system will see the start event first. Alternatively, you may program your system to simply ignore the first reading, but you will have to re-arm the event counters before the next occurrence of the start signal.

7.4 Other Comments

7.4.1 General-Purpose I/O

Any pins not in use for input capture can be used for general-purpose I/O.

7.4.2 Interrupts

If desired, an interrupt can be generated from either the counter rollover, a Begin signal, or an End signal. It can be used to service an interrupt that will get the value of CELR and CEMR when an end signal occurs.

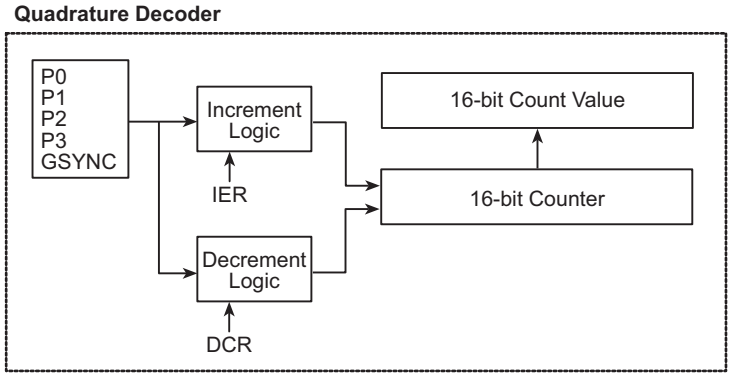
8. QUADRATURE DECODER

8.1 Overview

A quadrature decoder is a specialized up/down counter, where it is the phase relationship of the two inputs that determines the count direction. Quadrature decoders are widely used in applications requiring motion feedback.

The advantage of using the Rabbit RIO for this purpose is to free up resources that would otherwise be used by the processor to perform quadrature decoding. This is especially important for processors without this functionality built into the hardware.

8.1.1 Block Diagram

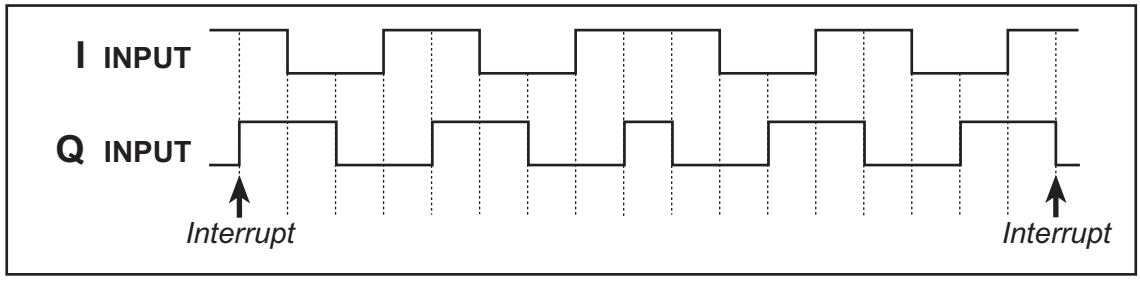


8.2 Dependencies

Item	Description	Register
Begin/End	Increment/decrement	ICR, DCR, CR
Pin	Defines pins as input	PnCR
Interrupt	Trigger Begin/End interrupt (optional)	IER, SR

8.3 Operation

Each Rabbit RIO I/O block provides one Quadrature Decoder channel. Each channel of the Quadrature Decoder accepts an in-phase (I) and a quadrature-phase (Q) signal. A source of the quadrature signal, such as a quadrature wheel, can be tracked in its motion by the relative phase of the I and Q signals. An example of such a signal is shown below where the quadrature signals first increment the counter, then decrement it. There are 18 counts



Choosing which pin is the in-phase signal and which pin is the quadrature-phase signal is arbitrary, but generally, these two signals should be connected such that the direction considered forward results in incrementing the counter and the backward direction should decrement the counter. In this case, incrementing and decrementing the counter depends on the transition of the I and Q inputs. For example, if the I and Q inputs were to transition from 10b to 11b to 01b and finally to 00b, the counter will increment as shown for the first two cycles of the waveform above. However, if the I and Q inputs were to transition from 01 to 11 to 10 and finally to 00, the counter will decrement as shown for the last two cycles of the waveform above.

Common quadrature wheels have either 500 or 512 steps, which would require count limits of either 0x7CF (1999) or 0x1FF (2047), as each step leads to a count of four, with each of the four counts corresponding to a rising or falling edges of the I and Q signals. One can get the quadrature count by reading the value from the 16-bit Count Value Registers. This is the typical use of the quadrature decoder.

8.3.1 Setup

The following steps explain how to set up a Quadrature Decoder channel.

1. Enable the counter mode in the Mode Register
2. Set the interrupts, if desired, in the Interrupt Enable Register
3. Define the in-phase signal and the quadrature signal in the Increment/In-Phase/Begin Control Register and the Decrement/Quadrature/End Control Register.

8.3.2 Example

The input signals in Figure 8-1 will increment the counter by 18 counts when the following registers are written.

Register	Value	Description
MR	0x03	Use master clock, enable counter mode for IIB and DQE.
ICR	0x24	Increment on quadrature transitions, use Pin 0 for in-phase signal.
DCR	0x25	Decrement on quadrature transitions, use Pin 1 for quadrature signal.
P0CR	0x00	Pin 0 an input.
P1CR	0x00	Pin 1 an input.

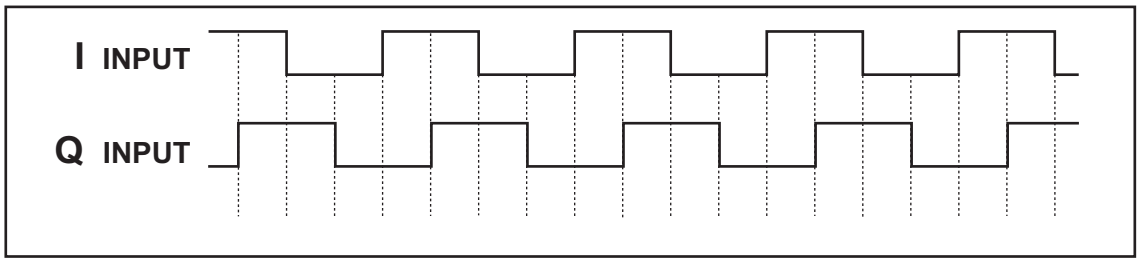


Figure 8-1. Sample Quadrature Input With Counter Incrementing

Similarly, the phase shift in Figure 8-2 will decrement the counter by 18. Keep in mind that each step, or complete transition cycle, leads to a count of four.

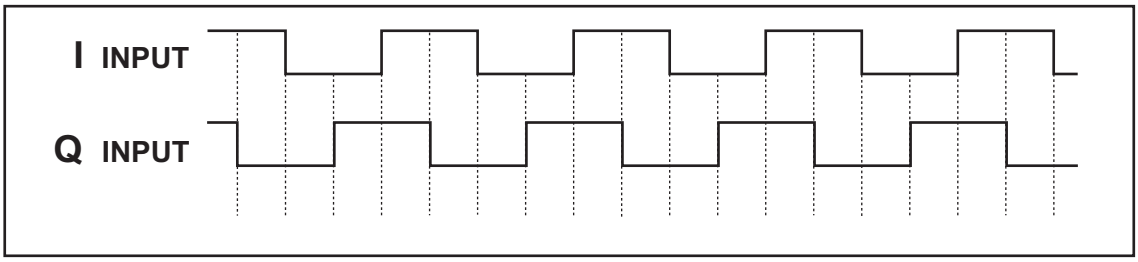


Figure 8-2. Sample Quadrature Input With Counter Decrementing

8.4 Other Comments

8.4.1 General-Purpose I/O

Any pins not in use for the quadrature decoder can be used for general-purpose I/O.

8.4.2 External Synchronization

As previously mentioned, the synch pin can be defined to reset the counter. Doing so can provide a point of reference from which to start counting so that accurate determination of position is possible when reading shaft rotation.

8.4.3 Interrupts

If desired, an interrupt can be generated from either the counter rollover, an in-phase signal, or a quadrature signal. It can be useful to service an interrupt that will get the value of CVLR and CVMR when an interrupt signal occurs. However, simple polling may be more effective since servicing interrupts every time the shaft moves may be a lot of overhead.

9. RABBITNET HUB

9.1 Overview

The Rabbit RIO may be used as a RabbitNet hub to provide a simple, efficient, and flexible means of establishing a network of RabbitNet peripheral cards and Rabbit RIOs ("RabbitNet devices"). A RabbitNet hub works very differently from a RabbitNet device. The RabbitNet architecture allows a hub to connect up to seven downstream devices, and supports two levels of hubs to allow a master to control up to a total of 49 devices. The hub routes information to its correct destination, whether that is one RabbitNet device on the network or as many as 49 devices.

RabbitNet uses an SPI-like interface that includes three signals: CLK, MISO, and MOSI. CLK is the clock used to synchronize data going in or coming out of the master with another device. MISO (Master In Slave Out) is the data from the slave (such as a RabbitNet device) going into the master. MOSI (Master Out Slave In) is the data from the master going into the slave. CLK, MISO, MOSI, and /CS now occupy all four pins of each I/O block.

9.2 Hub Functions

A hub is responsible for switching its upstream port to one of its downstream ports, where a master is upstream and a RabbitNet device is downstream. The master selects a downstream port by writing the HOS (Hub Output Select) byte before the device CMD (Command). The information in CMD is the same information sent during an addressing cycle for the RabbitNet device. All RabbitNet device ignore the HOS byte, since it applies only to hubs to determine which port to route information to.

A hub demultiplexes the upstream /CS to one of the downstream /CSs. The same goes for MOSI and CLK. It also multiplexes one of the downstream MISOs to the upstream MISO. The downstream /CS will either be inactive, or it will reflect the state of the upstream /CS. No downstream /CS will be active when the upstream /CS is inactive.

9.3 Hub Commands

A hub only acts on the first byte it receives after /CS goes active. All the following bytes are ignored by the hub, but are passed through until the /CS goes inactive again. The exception to this is when the hub is itself addressed (as device 7). In this case, it acts like a normal device. This may occur if the hub's internal registers need to be accessed.

A HOS byte is similar to a CMD byte, except that the MSB of this byte is 1. When the MSB is set for the first byte seen by the hub after /CS is activated, the hub recognizes this as a HOS byte (or possibly a hub broadcast if the second MSB is also set). In this case, the following 6 bits are used to select the output from the first and second level hubs (0 to 7). For convenience, HOS commands use octal notation to refer to particular output combinations. For example, HOS46 selects Port 4 for the first-level hub, and Port 6 for the second-level hub.

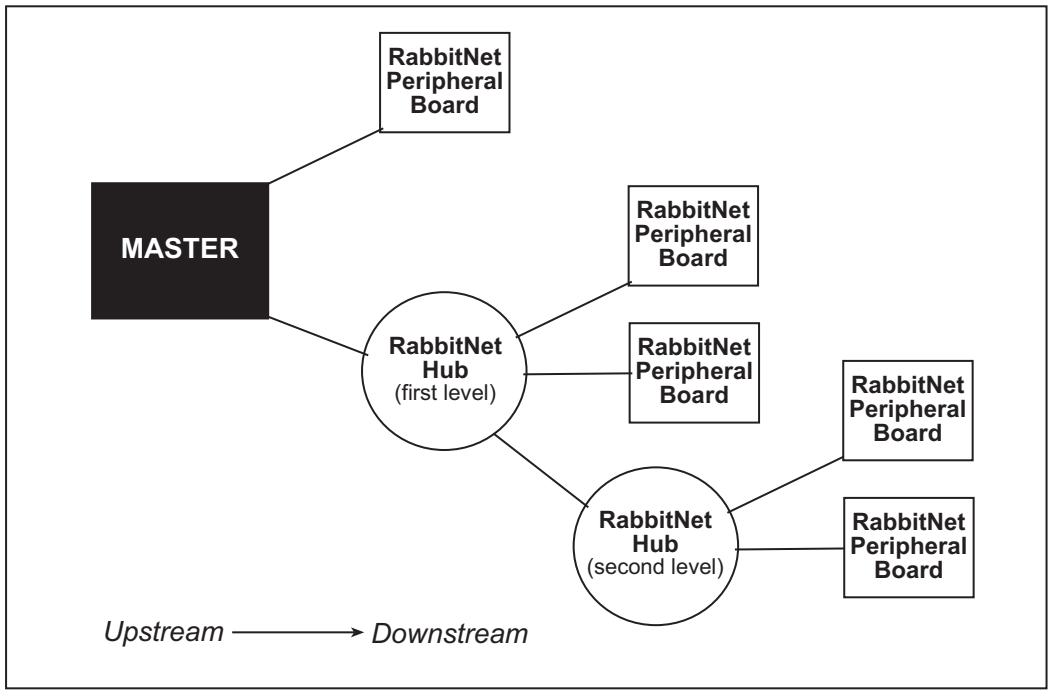


Figure 9-1. RabbitNet Network Structure

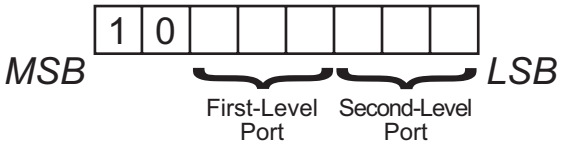


Figure 9-2. HOS Command Bit Field

In order to decode the output select field, a hub has to know whether it is a first- or second-level hub. A first-level hub uses bits 5-3, and a second-level hub uses bits 2-0. When a hub is reset (including a software reset from the master), it assumes that it is a second-level hub. Following reset, all the downstream ports are detected. The master sends out a distinctive HOS77 when it is enumerating the network after a reset. Only a first-level hub can possibly see this command, so any hub that sees HOS77 promotes itself to the first level. An output select field containing 7 indicates the hub itself.

Hubs remember the most recent HOS command. This allows the host to omit the HOS on subsequent transactions when accessing the same device as before. In other words, the port selection doesn’t change until a different HOS command is encountered — this is signalled when the MSB is a "1" in the first byte following the assertion of a chip select. Accessing a second-level device requires two HOS commands to be sent out, one for the first-level hub and one for the second-level hub.

9.4 Reset and Enumeration

When the master is reset, or at any other time that it becomes necessary, the entire network may be reset. This means that all attached devices and hubs transition to a known initial state. Following reset, the master probes the network for accessible devices and hubs. This is known as network enumeration. This process allows the master to construct a mapping of devices and tier ID information. After enumeration, the network is ready for application data.

9.5 Additional RabbitNet Information

Additional information on RabbitNet and the RabbitNet peripheral cards is available in the ***RabbitNet Peripheral Card User’s Manual***.

9.6 Registers

The RabbitNet hub requires G//B to be *low*, /P/I to be *high*, and SER//PAR to be *high*.

RNA[5:0]	Selects
000000	RabbitNet parameters (0x00)
000001	Product ID (0x10)
000010	Reserved (0x00)
000011	Reserved (0x00)
000100	Reserved (0x00)
000101	Reserved (0x00)
000110	Reserved (0x00)
000111	Reset Status (0x00)
001xxx	Reserved (0x00)
01xxxx	Reserved (0x00)
1xxxxx	Rabbit RIO “External Address”

9.6.1 RabbitNet Status Register

The RabbitNet Status Register holds the information about a Rabbit RIO's function in a RabbitNet network. This byte is automatically sent during a HOS or CMD command.

RabbitNet Status Register (RNSR) (RN Address = none)		
Bit(s)	**Value**	**Description**
7:6		These bits always return 01 when read.
5	0	This is a RabbitNet device.
	1	This is a RabbitNet hub.
4		This bit always returns 0 when read.
3	0	This is a RabbitNet device or a RabbitNet Level 2 hub.
	1	This is a RabbitNet Level 1 hub.
2	0	There is no interrupt pending.
	1	There is an interrupt pending.
1		This bit always returns 0 when read.
0	0	This RabbitNet device or hub is not in the "fail-safe" mode.
	1	This RabbitNet device or hub is in the "fail-safe" mode. The "fail-safe" mode is entered by a hardware reset, the hard reset command, or by a watchdog timeout. While in the "fail-safe" mode only writes to the RabbitNet Reset Status Register are accepted.

9.6.2 RabbitNet Parameter Register

RabbitNet Parameter Register (RNPR) (RN Address = 0x00)		
Bit(s)	**Value**	**Description**
7:0	Read	This RabbitNet address always returns 0x00 when read.

9.6.3 RabbitNet ID Register

During enumeration, this register can be read in by the master to determine the function of this device.

RabbitNet ID Register (RNIDR) (RN Address = 0x01)		
Bit(s)	**Value**	**Description**
7:0	Read	This RabbitNet address returns 0x10 for a hub and 0xF0 for a device.

9.6.4 RabbitNet Reset Status Register

The RabbitNet Reset Status Register allows a master to reset the device.

RabbitNet Reset Status Register (RNIDR) (RN Address = 0x07)		
Bit(s)	**Value**	**Description**
7	0	No operation. This bit always returns 0 when read.
	1	Perform a hard reset. This command is equivalent to a hardware reset.
6	0	No operation. This bit always returns 0 when read.
	1	Perform a soft reset. This is the only command that resets the “fail-safe” bit.
5:0		These bits are ignored during write and always return zeros during read.

INDEX

R

S

T

V