PRODUCT MANUAL

Rabbit® 3000 Microprocessor

User's Manual

019-0108_Z

Rabbit 3000 Microprocessor User's Manual

The latest revision of this manual is available at www.digi.com.

TABLE OF CONTENTS

1. THE RABBIT 3000 PROCESSOR

1.1 Introduction

Rabbit Semiconductor was formed expressly to design a better microprocessor for use in small- and medium-scale single-board computers. The first microprocessors was the *Rabbit 2000*. Besides the *Rabbit 3000*, *Rabbit 4000* and *Rabbit 5000* microprocessors are also available. Rabbit microprocessor designers have had years of experience using Z80, Z180, and HD64180 microprocessors in small single-board computers. The Rabbit microprocessors share a similar architecture and a high degree of compatibility with these microprocessors, but represent a vast improvement.

The Rabbit 3000 is a high-performance microprocessor with low electromagnetic interference (EMI), and is designed specifically for embedded control, and communications. The 8-bit Rabbit 3000 outperforms most 16–bit processors without losing the efficiency of an 8–bit architecture. Extensive integrated features and glueless architecture facilitate rapid hardware design, while a C-friendly instruction set promotes efficient development of even the most complex applications.

The Rabbit 3000 is fast, running at up to 55.5 MHz, with compact code and support for up to 6 MB of memory. Operating with a 1.8 V to 3.6 V power supply, the Rabbit 3000 boasts six serial ports with IrDA, 56+ digital I/O, Quadrature Decoder, PWM outputs, and pulse capture and measurement capabilities. It also features a battery-backable real-time clock, glueless memory and I/O interfacing, and ultra-low power modes. Four levels of interrupt priority allow fast response to real-time events. Its compact instruction set and high clock speeds give the Rabbit 3000 exceptionally fast math, logic, and I/O performance.

1.2 Features

The Rabbit 3000 has several powerful design features that practically eliminate EMI problems, which is essential for OEMs who need to pass CE and regulatory radiofrequency emissions tests. The amplitude of any electromagnetic radiation is reduced by the internal spectrum spreader, by gated clocks (which prevent unnecessary clocking of unused registers), and by separate power planes for the processor core and I/O pins (which reduce noise crosstalk). An external I/O bus can be used by designers to enable separate buses for I/O and memory or to limit loading the memory bus to reduce EMI and ground bounce problems when interfacing external peripherals to the processor. The external I/O bus accomplishes this by duplicating the Rabbit's data bus on Parallel Port A, and uses Parallel Port B to provide the processor's six or eight least significant address lines for interfacing with external peripherals.

The high-performance instruction set offers both greater efficiency and execution speed of compiler-generated C code. Instructions include numerous single-byte opcodes that execute in two clock cycles, 16-bit and 32-bit loads and stores, 16-bit and 32-bit logical and arithmetic operations, 16 × 16 multiply (executes in 12 clocks), long jumps and returns for accessing a full 6 MB of memory, and one-byte prefixes to turn memory-access instructions into internal and external I/O instructions. Hardware-supported breakpoints ease debugging by trapping on code execution or data reads and writes.

The Rabbit 3000 requires no external memory driver or interface-logic. Its 20-bit address bus, 8-bit data bus, three chip-select lines, two output-enable lines, and two write-enable lines can be interfaced directly with up to six memory devices. Up to 1 MB of memory can be accessed directly via the Dynamic C development software, and up to 6 MB can be interfaced with additional software development. A built-in slave port allows the Rabbit 3000 to be used as master or slave in multi-processor systems, permitting separate tasks to be assigned to dedicated processors. An 8-line data port and five control signals simplify the exchange of data between devices. A remote cold boot enables startup and programming via a serial port or the slave port.

The Rabbit 3000 features seven 8-bit parallel ports, yielding a total of 56 digital I/O. Six CMOS-compatible serial ports are available. All six are configurable as asynchronous (including output pulses in IrDA format), while four are configurable as clocked serial (SPI) and two are configurable as SDLC/HDLC. The various internal peripherals share the parallel port I/O pins.

The Rabbit 3000 also offers many specialized peripherals. Two Input Capture channels each have a 16-bit counter, clocked by the output of an internal timer, that can be used to capture and measure pulses. These measurements can be extended to a variety of functions such as measuring pulse widths or for baud-rate autodetection. Two Quadrature Decoder channels each have two inputs, as well as an 8-bit up/down counter. Each Quadrature Decoder channel provides a direct interface to optical encoder units. Four independent pulse-width modulator (PWM) outputs, each based on a 1024-pulse frame, are driven by the output of a programmable internal timer. The PWM outputs can be filtered to create a 10-bit D/A converter or they can be used directly to drive devices such as motors or solenoids.

There are numerous timers available for use in the Rabbit 3000. Timer A consists of ten 8-bit counters, each of which has a programmed time constant. Six of them can be cascaded from the primary Timer A counter. Timer B contains a 10-bit counter, two match registers, and two step registers. An interrupt can be generated or the output pin can be updated when the counter reaches a match value, and the match value is then incremented automatically by the step value.

The Rabbit 3000 (Rev. A and later versions) also provides support for protected operating systems. Support for two levels of operation, known as *system* and *user* modes, allow application-critical code to operate in safety while user code is prevented from inadvertently disturbing the setup of the processor. Memory blocks as small as 4 KB can be write-protected against accidental writes by user code, and stack over/underflows can be trapped by high-priority interrupts.

1.3 Block Diagram

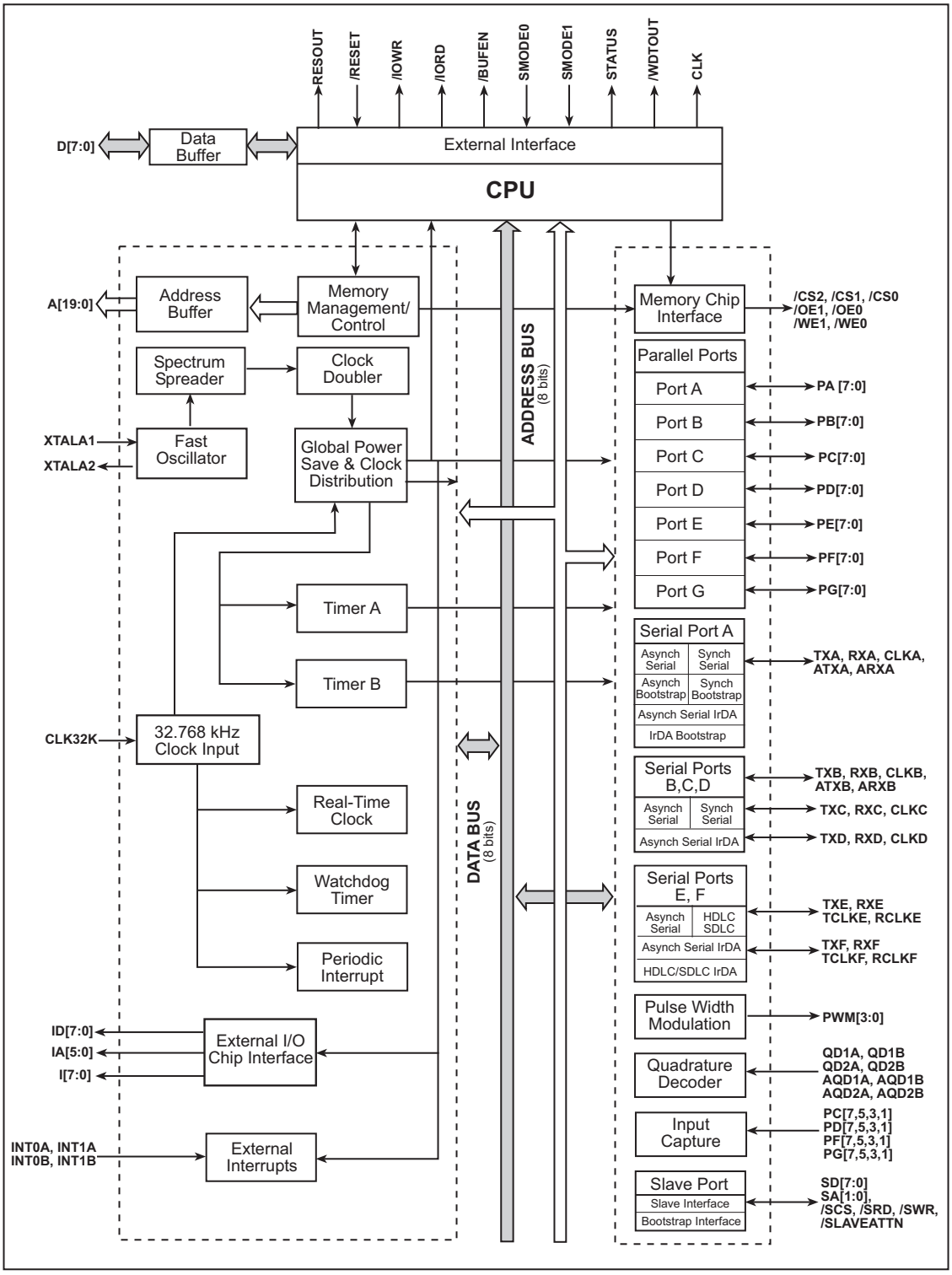


Figure 1-1. Rabbit 3000 Block Diagram

1.4 Basic Specifications

Table 1-1. Rabbit 3000 Specifications and Features

Package	128-pin LQFP	128-ball TFBGA
Package Size	16 mm × 16 mm × 1.5 mm	10 mm × 10 mm × 1.2 mm
Operating Voltage	1.8–3.6 V DC	
Operating Current	2 mA/MHz @ 3.3 V	
Operating Temp.	-55°C to +85°C	
Maximum Clock Speed	55.5 MHz	
Digital I/O	56+ (arranged in seven 8-bit ports)	
Serial Ports	6 CMOS-compatible	
Baud Rate	Clock speed/8 max. asynchronous	
Address Bus	20-bit	
Data Bus	8-bit	
Timers	Ten 8-bit and one 10-bit with 2 match registers	
Real-Time Clock	Yes, battery backable	
RTC Oscillator Circuitry	External	
Watchdog Timer/Supervisor	Yes	
Clock Modes	1×, 2×, /2, /3, /4, /6, /8	
Power-Down Modes	Sleepy (32 kHz) Ultra-Sleepy (16, 8, 2 kHz)	
External I/O Bus	8 data, 8 address lines	

1.5 Comparing Rabbit Microprocessors

The Rabbit 2000, Rabbit 3000, Rabbit 4000, and Rabbit 5000 features are compared below.

Feature	Rabbit 5000	Rabbit 4000	Rabbit 3000	Rabbit 2000
Maximum Clock Speed, industrial Maximum Clock Speed, commercial	100 MHz 100 MHz	60 MHz 60 MHz	55.5 MHz 58.8 MHz	30 MHz 30 MHz
Maximum Crystal Frequency Main Oscillator (may be doubled internally up to maximum clock speed)	100 MHz	60 MHz	30 MHz	30 MHz
32.768 kHz Crystal Oscillator	External	External	External	Internal
Operating Voltage, core Operation Voltage, I/O	1.8 V ± 10% 3.3 V or 1.8 V ± 10%	1.8 V ± 10% 3.3 V or 1.8 V ± 10%	3.3 V ± 10%	5.0 V ± 10%
Maximum I/O Input Voltage	3.6 V	3.6 V	5.5 V	5.5 V
Current Consumption	0.57 mA/MHz @ 1.8 V/3.3 V (Wi-Fi and Ethernet disabled)	0.35 mA/MHz @ 3.3 V	2 mA/MHz @ 3.3 V	4 mA/MHz @ 5 V
Number of Package Pins	289/196	128	128	100
Size of Package, LQFP/PQFP Spacing Between Package Pins	N/A	16 × 16 × 1.5 mm 0.4 mm (16 mils)	16 × 16 × 1.5 mm 0.4 mm (16 mils)	24 × 18 × 3 mm 0.65 mm (26 mils)
Size of Package, BGA (mm) Spacing Between Package Pins	15 × 15 × 1.4 12 × 12 × 1.2 0.8 mm	10 × 10 × 1.2 0.8 mm	10 × 10 × 1.2 0.8 mm	Not available
Separate Power and Ground for I/O Buffers (EMI reduction)	Yes	Yes	Yes	No
Clock Spectrum Spreader	Yes	Yes	Yes	Rabbit 2000B/C
Clock Modes	1×, 2×, /2, /3, /4, /6, /8	1×, 2×, /2, /3, /4, /6, /8	1x, 2x, /2, /3 /4, /6, /8	1x, 2x, /4, /8
Powerdown Modes, sleepy Powerdown Modes, ultra sleepy	32 kHz 16, 8, 4, 2 kHz	32 kHz 16, 8, 4, 2 kHz	32 kHz 16, 8, 4, 2 kHz	32 kHz
Low-Power Memory Control	Short and Self-Timed Chip Selects	Short and Self-Timed Chip Selects	Short and Self-Timed Chip Selects	None
Extended Memory Timing for High-Frequency Operation	Yes	Yes	Yes	No
Number of 8-bit I/O Ports	6	5	7	5

Feature	Rabbit 5000	Rabbit 4000	Rabbit 3000	Rabbit 2000
External I/O Data/Address Bus	Yes	Yes	Yes	None
Number of Serial Ports	6	6	6	4
Serial Ports Capable of SPI/ Clocked Serial	4 (A, B, C, D)	4 (A, B, C, D)	4 (A, B, C, D)	2 (A, B)
Serial Ports Capable of SDLC/ HDLC	2 (E, F)	2 (E, F)	2 (E, F)	None
Asynch Serial Ports With Support for IrDA Communication	6	6	6	None
Serial Ports with Support for SDLC/HDLC IrDA Communication	2	2	2	None
Maximum Asynchronous Baud Rate	Clock Speed/8	Clock Speed/8	Clock Speed/8	Clock Speed/32
Ethernet Port	10/100Base-T	10Base-T	None	None
Input Capture Units	2	2	2	None
Quadrature Decoders	2 channels	2 channels	2 channels	None

Appendix B summarizes the issues and revisions made to the Rabbit 3000. Note that the original version of the Rabbit 3000 is no longer sold.

2. CLOCKS

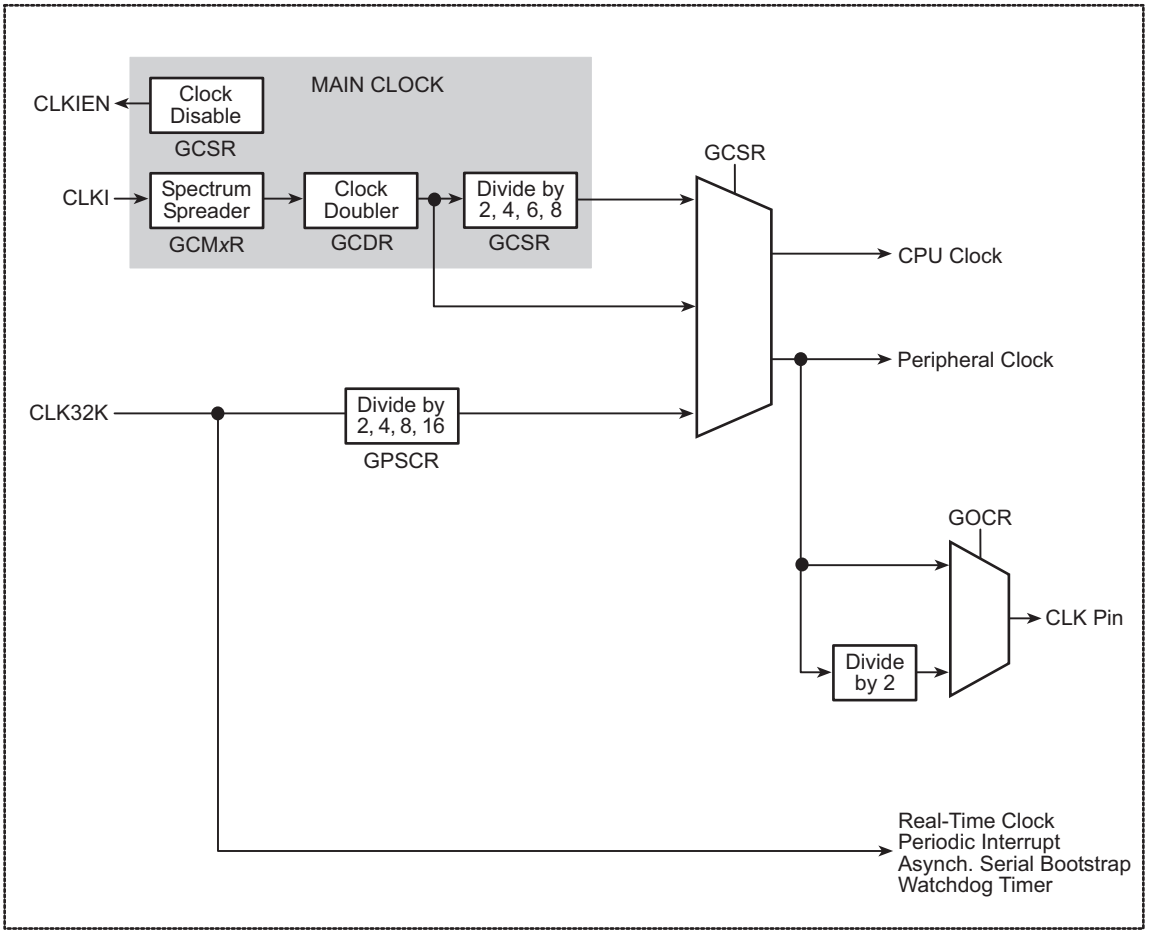
2.1 Overview

The Rabbit 3000 supports two separate clocks—the main clock and the 32 kHz clock. The main clock is used to derive the processor clock and the peripheral clock inside the processor. The 32 kHz clock is used to drive the asynchronous serial bootstrap, the real-time clock, the periodic interrupt, and the watchdog timers.

The Rabbit 3000 has a spectrum spreader on the main clock that shortens and lengthens clock cycles. This has the net effect of reducing the peak energy of clock harmonics by spreading the spectral energy into nearby frequencies, which reduces EMI and facilitates government-mandated EMI testing. Gated clocks are used whenever possible to avoid clocking unused portions of the processor, and separate power-supply pins for the core and I/O ring further reduce EMI from the Rabbit 3000.

The main clock can be doubled or divided by 2, 4, 6, or 8 to reduce EMI and power consumption. The 32 kHz clock (which can be divided by 2, 4, 8, or 16) can be used instead of the main clock to generate processor and peripheral clocks as low as 2 kHz for significant power savings. Note that dividing the 32 kHz clock only affects the processor and peripheral clocks; the full 32 kHz signal is still provided to the peripherals (RTC and watchdog timers) that use it directly. The periodic interrupt is automatically disabled since there is not enough time to process it when running off the 32 kHz clock.

2.1.1 Block Diagram



2.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
Global Control/Status Register	GCSR	0x0000	R/W	11000000	
Global Clock Modulator 0 Register	GCM0R	0x000A	W	00000000	
Global Clock Modulator 1 Register	GCM1R	0x000B	W	00000000	
Global Power Save Control Register	GPSCR	0x000D	W	0000x000	00000000
Global Clock Double Register	GCDR	0x000F	W	00000000	

2.2 Dependencies

2.2.1 I/O Pins

The main clock input is on the CLKI pin. There is an internal Schmitt trigger on this pin to remove problems with noise on slowly transitioning signals.

The main clock disable output is on the CLKIEN pin. Its state is changed by one of the bit combinations of bits 4:2 in GCSR.

The 32 kHz clock input is on the CLK32K pin. There is an internal Schmitt trigger on this pin as well.

The peripheral clock or peripheral clock divided by 2 may be optionally output on the CLK pin by enabling it via bits 7:6 in GOCR.

2.2.2 Other Registers

Register	Function
GOCR	Used to set up the CLK output pin.

2.3 Operation

2.3.1 Main Clock

The main clock is input on the CLKI pin, and is optionally sent through the spectrum spreader and then the clock doubler. Both of these are described in greater detail below.

Different main clock modes may be selected via the GCSR, as shown in Table 2-1. Note that one GCSR setting slows the processor clock while the peripheral clock operates at full speed; this allows some power reduction while keeping settings like serial baud rates and the PWM at their desired values.

Table 2-1. Clock Modes

GCSR Setting	Processor Clock	Peripheral Clock
xxx010xx	Main clock	Main clock
xxx011xx	Main clock / 2	Main clock / 2
xxx110xx	Main clock / 4	Main clock / 4
xxx111xx	Main clock / 6	Main clock / 6
xxx000xx	Main clock / 8	Main clock / 8 (default on startup)
xxx001xx	Main clock / 8	Main clock
xxx100xx	32 kHz clock (possibly divided)	32 kHz clock (possibly divided via GPSCR)
xxx101xx	32 kHz clock (possibly divided); main clock disabled via CLKIEN output signal	32 kHz clock (possibly divided via GPSCR)

When the 32 kHz clock is enabled in GCSR, it can be further divided by 2, 4, 6, or 8 to generate even lower frequencies by enabling those modes in bits 0–2 of GPSCR. See Table 2-4 for more details.

2.3.2 Spectrum Spreader

When enabled, the spectrum spreader stretches and compresses the main clock in a complex pattern that spreads the energy of the clock harmonics over a wider range of frequencies.

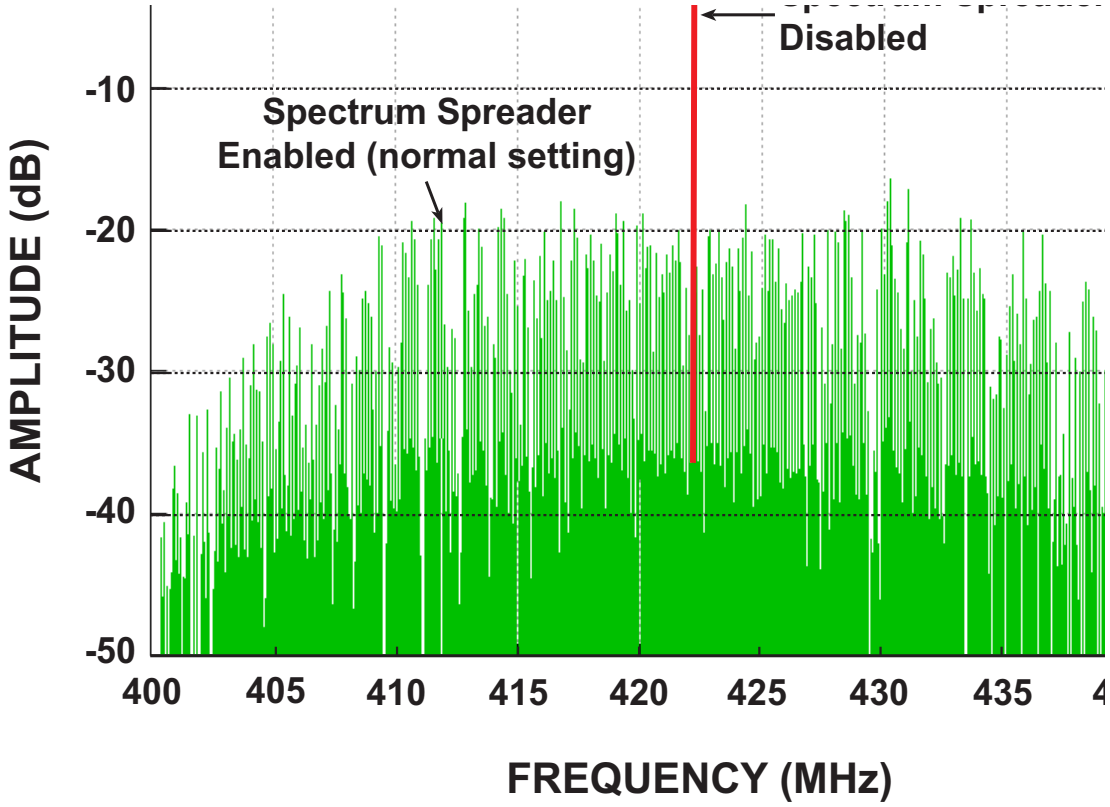


Figure 2-1. Effects of Spectrum Spreader

There are three settings that correspond to normal and strong spreading in the 0–50 MHz and >50 MHz main clock range. Each setting will affect the clock cycle differently; the maximum cycle shortening (at 1.8 V and 25°C) is shown in Table 2-2 below.

Table 2-2. Spectrum Spreader Settings

0–50 MHz	> 50 MHz	GCM0R Value	Description	Max. Cycle Shortening
—	Normal	0x0040	Normal spreading of frequencies over 50 MHz	2.3 ns
Normal	Strong	0x0000	Normal spreading of frequencies up to 50 MHz; strong spreading of frequencies over 50 MHz	3 ns
Strong	—	0x0080	Strong spreading of frequencies up to 50 MHz; normal spreading of frequencies over 50 MHz	4.5 ns

The spectrum spreader either stretches or shrinks the low plateau of the clock by a maximum of 3 ns for the normal spreading and up to 5 ns for the strong spreading. If the clock doubler is used, this will cause an additional asymmetry between alternate clock cycles.

Both normal and strong modes reduce clock harmonics by approximately 15 dB for frequencies above 100 MHz; for lower frequencies the strong setting has a greater effect in reducing the peak spectral strength as shown in Figure 2-2.

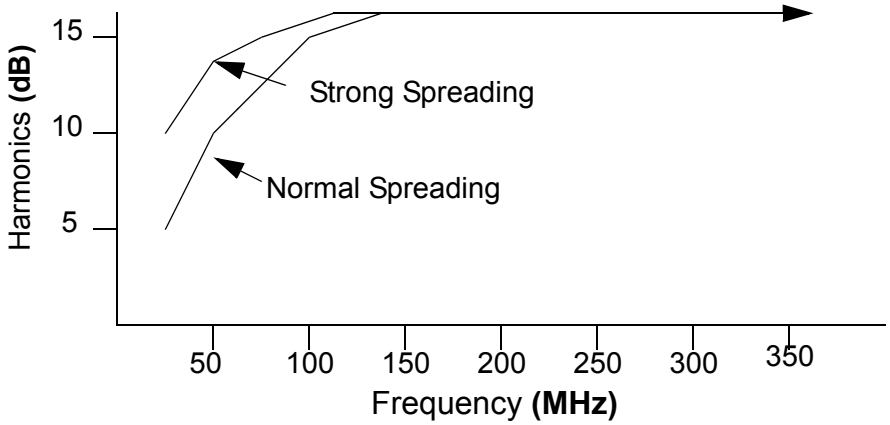


Figure 2-2. Peak Spectral Amplitude Reduction by Spectrum Spreader

Two registers control the clock spectrum spreader. These registers must be loaded in a specific manner with proper time delays. GCM0R is only read by the spectrum spreader at the moment when the spectrum spreader is enabled by storing 0x080 in GCM1R. If GCM1R is cleared (when disabling the spectrum spreader), there is up to a 500-clock delay before the spectrum spreader is actually disabled. The proper procedure is to clear GCM1R, wait for 500 clocks, set GCM0R, and then enable the spreader by storing 0x080 in GCM1R.

The spectrum spreader is applied to the main clock before the clock doubler, so if both are enabled there will be additional asymmetry between alternate clock cycles.If the clock doubler is used, the spectrum spreader affects every other cycle and reduces the clock high time. If the doubler is not used, then the spreader affects every clock cycle, and the clock low time is reduced.

2.3.3 Clock Doubler

The clock doubler allows a lower frequency crystal to be used for the main oscillator and to provide an added range over which the clock frequency can be adjusted. The clock doubler is controlled via the Global Clock Double Register (GCDR).

The clock doubler uses an on-chip delay circuit that must be programmed by the user at startup if there is a need to double the clock. Table 2-3 lists the recommended delays for the GCDR for various oscillator frequencies.

Table 2-3. Recommended Delays Set In GCDR for Clock Doubler

Recommended GCDR Value	Frequency Range
0x000F	≤7.3728 MHz
0x000B	7.3728–11.0592 MHz
0x0009	11.0592–16.5888 MHz
0x0006	16.5888–20.2752 MHz
0x0003	20.2752–52.8384 MHz
0x0000	>52.8384 MHz

When the clock doubler is used and there is no subsequent division of the clock, the output clock will be asymmetric, as shown in Figure 2-3.

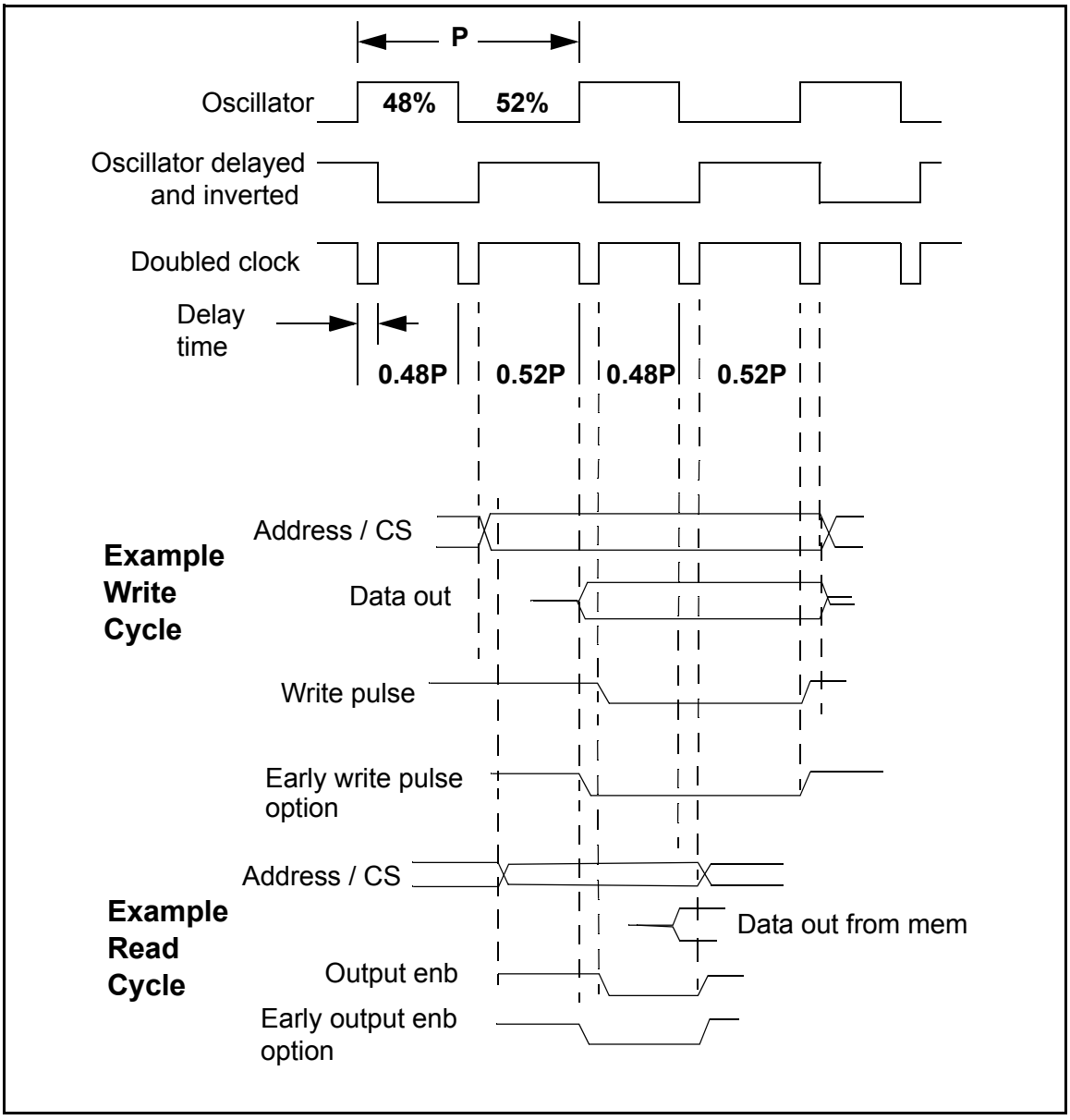


Figure 2-3. Effect of Clock Doubler

The doubled-clock low time is subject to wide (50%) variation since it depends on process parameters, temperature, and voltage. The times given above are for a core supply voltage of 3.3 V and a temperature of 25°C. The doubled-clock low time increases by 20% when the voltage is reduced to 2.5 V, and increases by about 40% when the voltage is reduced further to 2.0 V. The values increase or decrease by 1% for each 5°C increase or decrease in temperature. The doubled clock is created by xor'ing the delayed and inverted clock with itself. If the original clock does not have a 50-50 duty cycle, then alternate clocks will have a slightly different length. Since the duty cycle of the built-in oscillator can be as

asymmetric as 52% / 48%, the clock generated by the clock doubler will exhibit up to a 4% variation in period on alternate clocks. Memory access time is not affected because the memory bus cycle is 2 clocks long and includes both a long and a short clock, resulting in no net change due to asymmetry. However, if an odd number of wait states is used, then the memory access time will be affected slightly

The maximum allowed clock speed must be slightly reduced if the clock is supplied via the clock doubler. The only signals clocked on the falling edge of the clock are the memory and I/O write pulses and the early option memory output enable. See Chapter 5 for more information on the early output enable and write enable options.

The power consumption is proportional to the clock frequency, and for this reason power can be reduced by slowing the clock when less computing activity is taking place. The clock doubler provides a convenient method of temporarily speeding up or slowing down the clock as part of a power management scheme.

2.3.4 32 kHz Clock

The 32.768 kHz clock is used to drive the asynchronous serial bootstrap, the real-time clock, the periodic interrupt, and the watchdog timers. If these features are not used in a design, the use of the 32 kHz clock is optional.

A simplified version of the recommended oscillator circuit for the Rabbit 3000 is shown below. The values of resistors and capacitors may need to be adjusted for various frequencies and crystal load capacitances. Rabbit's Technical Note TN235, "External 32.768 kHz Oscillator Circuits," is available on the Rabbit Web site and goes into this circuit in detail.

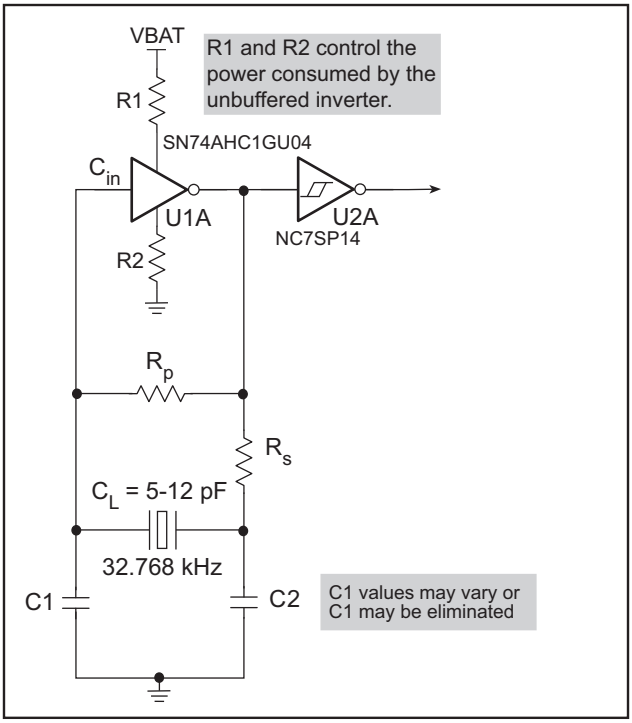


Figure 2-4. Basic 32.768 kHz Oscillator Circuit

The 32.768 kHz circuit consumes microampere level currents and has a very high impedance, making it susceptible to noise, moisture, and environmental contaminants. It is strongly recommended to conformally coat this circuit to limit effects of temperature and humidity on the oscillation frequency. Details about this requirement are available in Rabbit's Technical Note TN303, "Conformal Coating", from the Rabbit Web site.

The 32.768 kHz oscillator is slow to start oscillating after power-on. For this reason, a wait loop in the BIOS waits until this oscillator is oscillating regularly before continuing the startup procedure. If the clock is battery-backed, there will be no startup delay since the oscillator is already oscillating. The startup delay may be as much as 5 seconds. Crystals with low series resistance ($R < 35\ k\Omega$) will start faster.

The 32 kHz oscillator can be used to drive the processor and peripheral clock to provide significant power savings in “ultra-sleepy” modes. The 32 kHz oscillator can be divided by 2, 4, 8, or 16 to provide clock speeds as low as 2.048 kHz. Special self-timed chip selects are available to keep the memory devices enabled for as short a time as possible when an ultra-sleepy mode is enabled; see Chapter 25 for more details on reducing power consumption.

Table 2-4. Ultra-Sleepy Clock Modes

GPSCR Setting	Processor and Peripheral Clock
xxxxx000	32.768 kHz
xxxxx100	16.384 kHz
xxxxx101	8.192 kHz
xxxxx110	4.096 kHz
xxxxx111	2.048 kHz

When the 32 kHz clock is enabled, the periodic interrupt is disabled automatically. The real-time clock and watchdog timers keep running, and use the full 32 kHz clock even when the processor and peripheral clocks use a divider on the 32 kHz clock.

2.4 Register Descriptions

Global Control/Status Register (GCSR) (Address = 0x0000)		
Bit(s)	**Value**	**Description**
7:6 (Read-only)	00	No reset or watchdog timer timeout since the last read.
	01	The watchdog timer timed out. These bits are cleared by a read of this register.
	10	This bit combination is not possible.
	11	Reset occurred. These bits are cleared by a read of this register.
5	0	No effect on the periodic interrupt. This bit will always be read as zero.
	1	Force a periodic interrupt to be pending.
4:2	xxx	See table below to decode of this field.
1:0	00	Periodic interrupts are disabled.
	01	Periodic interrupts use Interrupt Priority 1.
	10	Periodic interrupts use Interrupt Priority 2.
	11	Periodic interrupts use Interrupt Priority 3.

GCSR Clock Select Field

Clock Select Bits 4:2 GCSR	CPU Clock	Peripheral Clock	Main Oscillator	Power-Save CS if Enabled by GPSCR
000	osc/8	osc/8	on	short CS option
001	osc/8	osc	on	short CS option
010	osc	osc	on	none
011	osc/2	osc/2	on	short CS option
100	32 kHz or fraction	32 kHz or fraction	on	self-timed option short CS option
101	32 kHz or fraction	32KHz or fraction	off	self-timed option short CS option
110	osc/4	osc/4	on	short CS option
111	osc/6	osc/6	on	short CS option

Global Clock Modulator 0 Register (GCM0R) (Address = 0x000A)		
Bit(s)	**Value**	**Description**
7	0	Enable normal spectrum spreading.
	1	Enable strong spectrum spreading.
6:0		These bits are reserved and should be written with zeros.

Global Clock Modulator 1 Register (GCM1R) (Address = 0x000B)		
Bit(s)	**Value**	**Description**
7	0	Disable the spectrum spreader.
	1	Enable the spectrum spreader.
6:0		These bits are reserved and should be written with zeros.

Global Power Save Control Register (GPSCR) (Address = 0x000D)		
Bit(s)	**Value**	**Description**
7:5	000	Self-timed chip selects are disabled.
	001	This bit combination is reserved and should not be used.
	011	This bit combination is reserved and should not be used.
	100	296 ns self-timed chip selects (192 ns best case, 457 ns worst case).
	101	234 ns self-timed chip selects (151 ns best case, 360 ns worst case).
	110	171 ns self-timed chip selects (111 ns best case, 264 ns worst case).
	111	109 ns self-timed chip selects (71 ns best case, 168 ns worst case).
4	0	Normal Chip Select operation.
	1	Short Chip Select timing when dividing main oscillator by 4, 6, or 8.
3* (Rabbit 3000A)	0	Normal Chip Select timing for write cycles
	1	Short Chip Select timing for write cycles (not available in full speed).
2:0	000	The 32 kHz clock divider is disabled.
	001	This bit combination is reserved and should not be used.
	011	This bit combination is reserved and should not be used.
	100	32 kHz oscillator divided by two (16.384 kHz).
	101	32 kHz oscillator divided by four (8.192 kHz).
	110	32 kHz oscillator divided by eight (4.096 kHz).
	111	32 kHz oscillator divided by sixteen (2.048 kHz).

* Bit 3 was always written with zero in the original Rabbit 3000 chip.

Global Clock Double Register (GCDR) (Address = 0x000F)		
Bit(s)	**Value**	**Description**
7:4		These bits are reserved and should be written with zeros.
3:0	0000	The clock doubler circuit is disabled.
	0001	6 ns nominal low time
	0010	7 ns nominal low time
	0011	8 ns nominal low time
	0100	9 ns nominal low time
	0101	10 ns nominal low time
	0110	11 ns nominal low time
	0111	12 ns nominal low time
	1000	13 ns nominal low time
	1001	14 ns nominal low time
	1010	15 ns nominal low time
	1011	16 ns nominal low time
	1100	17 ns nominal low time
	1101	18 ns nominal low time
	1110	19 ns nominal low time.
	1111	20 ns nominal low time

Global Output Control Register (GOCR) (Address = 0x000E)		
Bit(s)	**Value**	**Description**
7:6	00	CLK pin is driven by perclk.
	01	CLK pin is driven by perclk/2.
	10	CLK pin is low.
	11	CLK pin is high.
5:4	00	STATUS pin is active (low) during a first opcode byte fetch.
	01	STATUS pin is active (low) during an interrupt acknowledge.
	10	STATUS pin is low.
	11	STATUS pin is high.
3	1	WDTOUTB pin is low (1 cycle minimum, 2 cycles maximum, of 32 kHz).
	0	WDTOUTB pin follows watchdog function.
2	x	This bit is ignored.
1:0	00	/BUFEN pin is active (low) during external I/O cycles.
	01	/BUFEN pin is active (low) during data memory accesses.
	10	/BUFEN pin is low.
	11	/BUFEN pin is high.

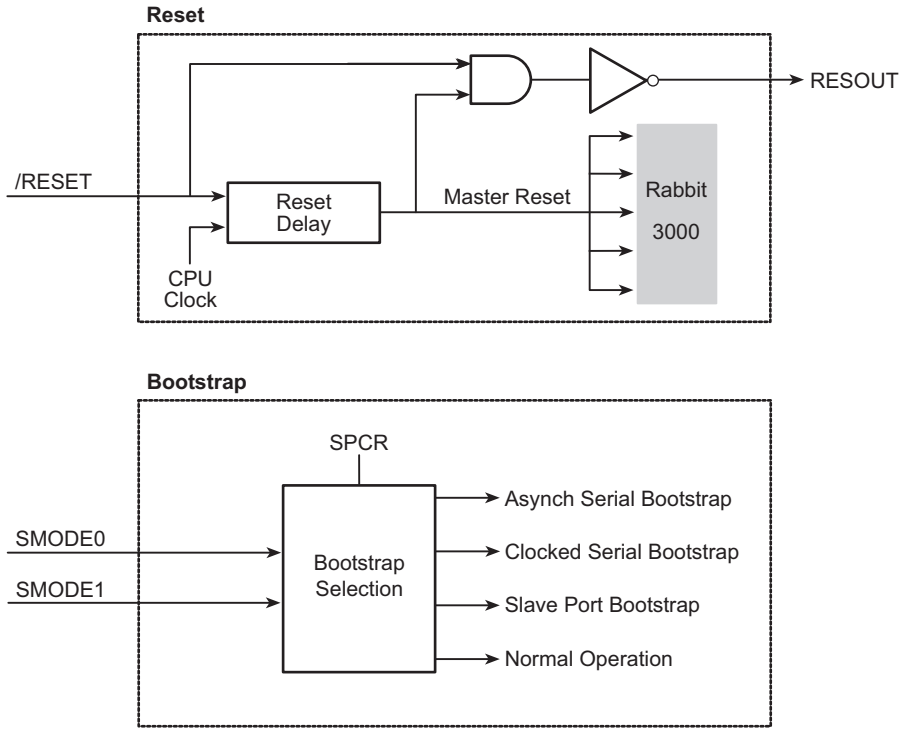
3. RESET AND BOOTSTRAP

3.1 Overview

The Rabbit 3000's /RESET pin initializes everything in the processor except for the real-time clock registers. If a write cycle is in progress, it waits until the write cycle is completed to avoid potential memory corruption.

After reset, the Rabbit 3000 checks the state of the SMODE pins. Depending on their state, it either begins normal operation by fetching instruction bytes from /CS0 and /OE0, or it enters a special bootstrap mode where it fetches bytes from either Serial Port A or the slave port. In this mode, bytes can be written to internal registers to set up the Rabbit 3000 for a particular configuration, or to memory to load a program. The processor can begin normal operation once the bootstrap operation is completed.

3.1.1 Block Diagram



3.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Slave Port Control Register	SPCR	0x0024	R/W	0xx00000

3.2 Dependencies

3.2.1 I/O Pins

SMODE0, SMODE1 — When the Rabbit 3000 is first powered up or when it is reset, the state of the SMODE0 and SMODE1 pins controls its operation.

/RESET — Pulling the /RESET pin low will initialize everything in the Rabbit 3000 except for the real-time clock registers.

/CS1 — During reset the impedance of the /CS1 pin is high, and all other memory and I/O control signals are held high. The special behavior of /CS1 allows an external RAM to be powered by the same source as the VBATIO pin (which powers /CS1). In this case, a pul-lup resistor is required on /CS1 to keep the RAM deselected during powerdown.

RESOUT — The RESOUT pin is high during reset and powerdown, but low at all other times, and can be used to control an external power switch to disconnect VDDIO from VBATIO when the main power source is removed.

3.2.2 Clocks

The processor requires a 32 kHz clock input to generate the 2400 bps internal clock required for asynchronous serial bootstrap, which is used when booting via Dynamic C and the Rabbit Field Utility. No 32 kHz clock is required for either clocked serial or slave port bootstrap.

When the processor comes out of reset, the CPU clock and peripheral clocks are both in divide-by-8 mode.

3.2.3 Other Registers

Register	Function
SPCR	Enable/disable processor monitoring of SMODE pins; read current state of SMODE pins.

3.2.4 Interrupts

There are no interrupts associated with reset or bootstrap.

3.3 Operation

Pulling the /RESET pin low will initialize everything in the Rabbit 3000 except for the real-time clock registers. The reset of the Rabbit 3000 is delayed until the completion of any write cycles in progress; reset takes effect immediately when no write cycles are occurring. The reset sequence requires a minimum of 128 cycles of the main clock to complete in either case.

During reset, the impedance of the /CS1 pin is high and all other memory and I/O control signals are held high. The special behavior of /CS1 allows an external RAM to be powered by the same source as the VBATIO pin (which powers /CS1). In this case, a pullup resistor is required on /CS1 to keep the RAM deselected during powerdown. The RESOUT pin is high during reset and powerdown, but low at all other times, and can be used to control an external power switch to disconnect VDDIO from VBATIO when the main power source is removed.

Table 3-1 lists the condition of the processor after reset takes place. The state of all registers after reset is provided in the chapter describing the specific peripheral.

Table 3-1. Rabbit 3000 Condition After Reset

Function	Operation After Reset
CPU Clock, Peripheral Clock	Divide-by-8 mode
Clock Doubler	Disabled
Memory Bank 0 Control Register	/CS0, /OE0, write-protected, 4 wait states
Memory Advanced Control Register	8-bit interface
CPU Registers: PC, SP, IIR, EIR, SU, HTR	0x0000
Interrupt Priority (IP Register)	0xFF (Priority 3)
Watchdog Timer	Enabled (2 seconds)
Secondary Watchdog Timer	Disabled

The processor checks the SMODE pins after the /RESET signal is inactive. Table 3-2 summarizes what happens:

- If both SMODE pins are zero, the Rabbit 3000 begins fetching instructions from the memory device on /CS0 and /OE0.
- If either of the SMODE pins is high, the processor will enter the bootstrap mode and accept triplets from either Serial Port A or the slave port. It is good practice to place pulldown resistors on the SMODE pins to ensure proper operation of your design.

Table 3-2. SMODE Pin Settings

SMODE Pins [1,0]	Operation
00	No bootstrap; code is fetched from address 0x0000 on /CS0, /OE0.
01	Bootstrap from the slave port.
10	Bootstrap from Serial Port A, clocked mode.
11	Bootstrap from Serial Port A, asynchronous mode.

In bootstrap mode, the processor inhibits the normal memory fetch from /CS0 and instead fetches instructions from a small internal boot ROM. This program reads triplets of three bytes from the selected peripheral. The first byte is the most-significant byte of a 16-bit address, the second byte is the least-significant byte of the address, and the third byte is the data to be written. If the uppermost bit of the address is 1, then the address is assumed to be an internal register address instead of a memory address, and the data are written to the appropriate register instead.

The boot ROM program waits for data to be available; each byte received automatically resets the watchdog timer with a 2-second timeout. Bytes must be received quickly enough to prevent timeout (or the watchdog must be disabled).

The device checks the state of the SMODE pins each time it jumps back to the start of the ROM program and responds according to the current state. In addition, by writing to bit 7 of the Slave Port Control Register (SPCR) the processor can be told to ignore the state of the SMODE pins and continue normal operation.

Note that the processor can be told to reenter bootstrap mode at any time by setting bit 7 of SPCR low; once this occurs and the least-significant four bits of the current PC address are zero, the processor will sample the state of the SMODE pins and respond accordingly. This feature allows in-line downloading from the selected bootstrap port; once the download is complete, bit 7 of SPCR can be set high and the processor will continue operating from where it left off.

As a security feature, any attempt to enter bootstrap mode from either the SMODE pins or by writing to bit 7 of SPCR will erase the data stored in the onchip-encryption RAM. This prevents loading a small program in memory to read out the data.

3.4 Register Descriptions

Slave Port Control Register (SPCR) (Address = 0x0024)		
Bit(s)	**Value**	**Description**
7	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
	Write	These bits are ignored and should be written with zero.
4	0	/SCS from PE7.
	1	/SCS from PB6*.
3:2 (Write only)	00	Disable the slave port. Parallel Port A is a byte-wide input port.
	01	Disable the slave port. Parallel Port A is a byte-wide output port.
	10	Enable the slave port.
	11	Enable the external I/O bus. Parallel Port A is used for the data bus and Parallel Port B[7:2] is used for the address bus.
1:0 (Write only)	00	Slave port interrupts are disabled.
	01	Slave port interrupts use Interrupt Priority 1.
	10	Slave port interrupts use Interrupt Priority 2.
	11	Slave port interrupts use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

4. SYSTEM MANAGEMENT

4.1 Overview

There are a number of basic system peripherals in the Rabbit 3000 processor, some of which are covered in later chapters. The peripherals covered in this chapter are the periodic interrupt, the real-time clock, the watchdog timers, and some of the miscellaneous output pins and their control and processor registers that provide the processor ID and revision numbers.

The periodic interrupt, when enabled, is generated every 16 clocks of the 32 kHz clock (every 488 µs, or 2.048 kHz). This interrupt can be used to perform periodic tasks.

The real-time clock (RTC) consists of a 48-bit counter that is clocked by the 32 kHz clock. It is powered by the VBAT pin, and so can be battery-backed. The value in the counter is not affected by reset, and can only be set to zero by writing to the RTC control register. The 48-bit width provides a 272-year span before rollover occurs.

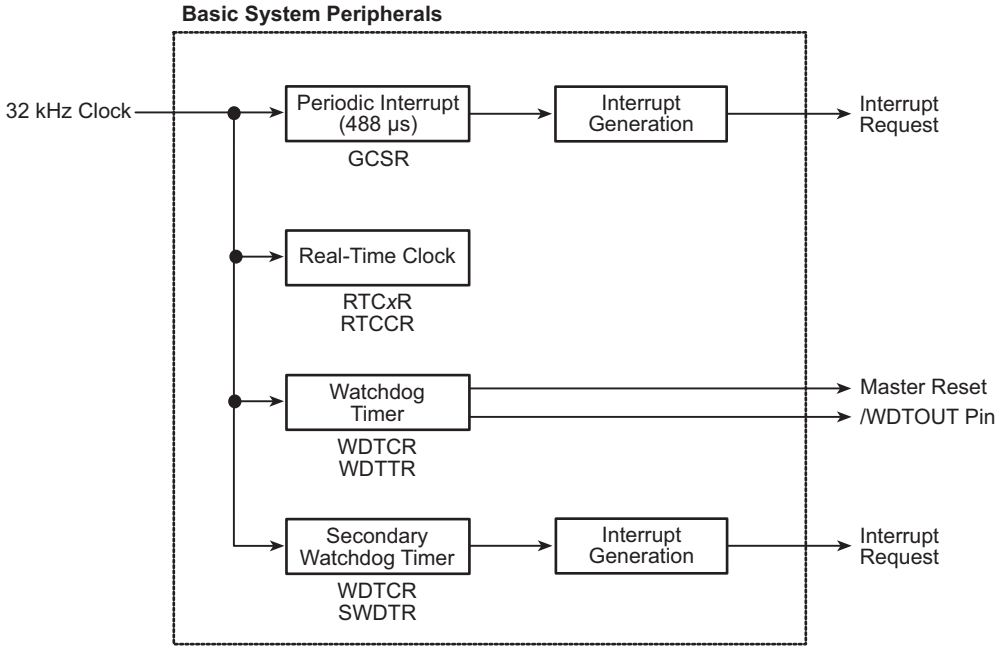
There are two watchdog timers in the Rabbit 3000, both clocked by the 32 kHz clock. The main watchdog timer can be set to time out from 250 ms to 2 seconds, and resets the processor if not reloaded within that time. Its purpose is to restart the processor when it detects that a program gets stuck or disabled.

The secondary watchdog timer, added in the Rabbit 3000A, can time out from 30.5 µs up to 7.8 ms, and generates a Priority 3 secondary watchdog interrupt when it is not reset within that time. The primary use for the secondary watchdog is to act as a safety net for the periodic interrupt — if the secondary watchdog is reloaded in the periodic interrupt, it will count down to zero if the periodic interrupt stops occurring. In addition, it can be used as a periodic interrupt on its own.

The following other registers are also described in this chapter.

- Global Output Control Register (GOCR), which controls the behavior of the CLK, STATUS, /WDT, and /BUFEN pins
- Global CPU Register (GCPU), which holds the identification number of the processor.
- Global Revision Register (GREV), which hold the revision number of the processor.

4.1.1 Block Diagram



4.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
Global Control/Status Register	GCSR	0x0000	R/W	11000000	
Real-Time Clock Control Register	RTCCR	0x0001	W	00000000	
Real-Time Clock Byte 0 Register	RTC0R	0x0002	R/W	xxxxxxxx	
Real-Time Clock Byte 1 Register	RTC1R	0x0003	R	xxxxxxxx	
Real-Time Clock Byte 2 Register	RTC2R	0x0004	R	xxxxxxxx	
Real-Time Clock Byte 3 Register	RTC3R	0x0005	R	xxxxxxxx	
Real-Time Clock Byte 4 Register	RTC4R	0x0006	R	xxxxxxxx	
Real-Time Clock Byte 5 Register	RTC5R	0x0007	R	xxxxxxxx	
Watchdog Timer Control Register	WDTCR	0x0008	W	00000000	
Watchdog Timer Test Register	WDTTR	0x0009	W	00000000	
Secondary Watchdog Timer Register	SWDTR	0x000C	W	—	11111111
Global Output Control Register	GOCR	0x000E	W	00000000	
Global ROM Configuration Register	GROM	0x002C	R	0xx00000	
Global RAM Configuration Register	GRAM	0x002D	R	0xx00000	
Global CPU Configuration Register	GCPU	0x002E	R	0xx00010	
Global Revision Register	GREV	0x002F	R	0xx00000	0xx00001

4.2 Dependencies

4.2.1 I/O Pins

The CLK, STATUS, /WDTOUT, and /BUFEN pins are controlled by GOCR. Each of these pins can be used as general-purpose outputs by driving them high or low:

- the CLK pin can output the peripheral clock, the peripheral clock divided by two, or be driven high or low;
- the STATUS pin can be active low during the first byte of each opcode fetch, active low during an interrupt acknowledge, or driven high or low;
- the /WDTOUT pin can be active low whenever the watchdog timer resets the device or driven low; and
- the /BUFEN pin can be active low during external I/O cycles, active low during data memory cycles, or driven high or low.

The values in the battery-backed onchip-encryption RAM bytes are cleared If the signal on the SMODE pins changes state.

4.2.2 Clocks

The periodic interrupt, real-time clock, watchdog timer, and secondary watchdog timer require the 32 kHz clock.

4.2.3 Interrupts

The periodic interrupt is enabled in GCSR, and will occur every 488 µs. It is cleared by reading GCSR. It can operate at Priority 1, 2, or 3.

The secondary watchdog interrupt will occur whenever the secondary watchdog is enabled and allowed to count down to zero. It is cleared by restarting the secondary watchdog by writing to WDTCR. The secondary watchdog interrupt always occurs at Priority 3.

4.3 Operation

4.3.1 Periodic Interrupt

The following steps explain how a periodic interrupt is used.

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Enable the periodic interrupt by writing to GCSR.
3. The interrupt request is cleared by reading from GCSR.

A sample interrupt handler is shown below.

```
periodic_isr::
   push af
   ioi ld a, (GCSR)    ; clear the interrupt request and get status

   ; handle any periodic tasks here

   pop af
   ipres
   ret
```

4.3.2 Real-Time Clock

The real-time clock consists of six 8-bit registers that together comprise a 48-bit value. The real-time clock is not synchronized to the read operation, so the least-significant bit should be read twice and checked for matching values; if the two reads do not match, then the real-time clock may have been updating during the read and should be read again.

Writing to RTC0R latches the current real-time clock value into the RTCxR holding registers, so the following sequence should be used to read the real-time clock.

1. Write any value to RTC0R and then read back a value from RTC0R.
2. Write a value to RTC0R again, and again read back a value from RTC0R.
3. If the two values do not match, repeat Step 2 until the last two readings are identical.
4. At this point, registers RTC1R through RTC6R can also be read and used.

Note that the periodic interrupt and the real-time clock are clocked by the same edge of the 32 kHz clock; if read from the periodic interrupt, the count is guaranteed to be stable and only needs to be read once (assuming it occurs within one clock of the 32 kHz clock).

The real-time clock can be reset by writing the sequence 0x0040 – 0x0080 to RTCCR. It can be reset and left in the byte increment mode by writing 0x0040 – 0x00C0 to RTCCR and then writing bytes repeatedly to RTCCR to increment the appropriate bytes of the real-time clock. The byte increment mode is disabled by writing 0x0000 to RTCCR.

4.3.3 Watchdog Timer

The watchdog timer is enabled on reset with a 2-second timeout. Unless specific data are written to WDTCR before that time expires, the processor will be reset. The watchdog timer can be disabled by writing a sequence of two bytes to WDTTR as described in the register description.

Table 4-1. Watchdog Timer Settings

WDTCR Value	Effect
0x005A	Restart watchdog timer with 2-second timeout.
0x0057	Restart watchdog timer with 1-second timeout.
0x0059	Restart watchdog timer with 500-millisecond timeout.
0x0053	Restart watchdog timer with 250-millisecond timeout.
0x005F	Restart the secondary watchdog timer.

The watchdog timer also contains a special test mode that speeds up the timeout period by clocking it with the peripheral clock instead of the 32 kHz clock. This mode can be enabled by writing to WDTTR.

4.3.4 Secondary Watchdog Timer (Rabbit 3000A)

The secondary watchdog timer is disabled on reset, unless the reset occurs because the primary watchdog timer times out while the secondary watchdog timer is enabled. The BIOS provided by Rabbit Semiconductor in Dynamic C avoids this bug by disabling the secondary watchdog on startup or reset by writing 0x005F to WDTCR. The following steps explain how to use the secondary watchdog timer.

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Write the desired timeout period to SWDTR. This also enables the secondary watchdog timer.
3. Restart the secondary watchdog timer by either writing the timeout period to SWDTR or writing 0x005F to WDTCR.

If the secondary watchdog timer counts down to zero, a Priority 3 secondary watchdog interrupt will occur. This interrupt request is cleared by writing a new timeout value to SWDTR. A sample interrupt handler is shown below.

```
secwd_isr::
   push af

   ; determine why the interrupt occurred and take appropriate action

   ld a, 0x40          ; timeout period of 0x40/32kHz = 1.95ms
   ioi ld (SWDTR), a    ; clear the interrupt request

   pop af
   ipres
   ret
```

4.4 Register Descriptions

Global Control/Status Register (GCSR) (Address = 0x0000)		
Bit(s)	**Value**	**Description**
7:6 (Read-only)	00	No reset or watchdog timer timeout since the last read.
	01	The watchdog timer timed out. These bits are cleared by a read of this register.
	10	This bit combination is not possible.
	11	Reset occurred. These bits are cleared by a read of this register.
5	0	No effect on the periodic interrupt. This bit will always be read as zero.
	1	Force a periodic interrupt to be pending.
4:2	xxx	See table below to decode of this field.
1:0	00	Periodic interrupts are disabled.
	01	Periodic interrupts use Interrupt Priority 1.
	10	Periodic interrupts use Interrupt Priority 2.
	11	Periodic interrupts use Interrupt Priority 3.

GCSR Clock Select Field

Clock Select Bits 4:2 GCSR	CPU Clock	Peripheral Clock	Main Oscillator	Power-Save CS if Enabled by GPSCR
000	osc/8	osc/8	on	short CS option
001	osc/8	osc	on	short CS option
010	osc	osc	on	none
011	osc/2	osc/2	on	short CS option
100	32 kHz or fraction	32 kHz or fraction	on	self-timed option short CS option
101	32 kHz or fraction	32KHz or fraction	off	self-timed option short CS option
110	osc/4	osc/4	on	short CS option
111	osc/6	osc/6	on	short CS option

Real-Time Clock Control Register (RTCCR) (Address = 0x0001)		
Bit(s)	**Value**	**Description**
7:0	0x0000	Writing a 0x0000 to the RTCCR has no effect on the RTC counter. However, depending on what the previous command was, writing a 0x0000 may either 1. disable the byte increment function or 2. cancel the RTC reset command If the 0x00C0 command is followed by a 0x0000 command, only the byte increment function will be disabled. The RTC reset will still take place.
	0x0040	Arm RTC for a reset with code 0x0080 or reset and byte increment function with code 0x00C0.
	0x0080	Resets all six bytes of the RTC counter to 0x0000 if preceeded by arm command 0x0040.
	0x00C0	Resets all six bytes of the RTC counter to 0x0000 and enters byte increment mode—precede this command with 0x0040 arm command.
7:6	01	This bit combination must be used with every byte increment write to increment clock(s) register corresponding to bit(s) set to "1". Example: 01001101 increments registers: 0, 2,3. The byte increment mode must be enabled. Storing 0x0000 cancels the byte increment mode.
5:0	0	No effect on the real-time clock counter.
	1	Increment the corresponding byte of the real-time clock counter.

Real-Time Clock x Register (RTC0R) (Address = 0x0002) (RTC1R) (Address = 0x0003) (RTC2R) (Address = 0x0004) (RTC3R) (Address = 0x0005) (RTC4R) (Address = 0x0006) (RTC5R) (Address = 0x0007)		
Bit(s)	**Value**	**Description**
7:0	Read	The current value of the 48-bit RTC holding register is returned.
	Write	Writing to the RTC0R transfers the current count of the RTC to six holding registers while the RTC continues counting.

Watchdog Timer Control Register (WDTCR) (Address = 0x0008)		
Bit(s)	**Value**	**Description**
7:0	0x005A	Restart the watchdog timer with a 2-second timeout period.
	0x0057	Restart the watchdog timer with a 1-second timeout period.
	0x0059	Restart the watchdog timer with a 500 ms timeout period.
	0x0053	Restart the watchdog timer with a 250 ms timeout period.
	0x005F	Restart the secondary watchdog timer (starting with Rabbit 3000A chip).
	other	No effect on watchdog timer or secondary watchdog timer.

Watchdog Timer Test Register (WDTTR) (Address = 0x0009)		
Bit(s)	**Value**	**Description**
7:0	0x0051	Clock the least significant byte of the watchdog timer from the peripheral clock. (Intended for chip test and code 0x0054 below only.)
	0x0052	Clock the most significant byte of the watchdog timer from the peripheral clock. (Intended for chip test and code 0x0054 below only.)
	0x0053	Clock both bytes of the watchdog timer, in parallel, from the peripheral clock. (Intended for chip test and code 0x0054 below only.)
	0x0054	Disable the watchdog timer. This value, by itself, does not disable the watchdog timer. Only a sequence of two writes, where the first write is 0x0051, 0x0052, or 0x0053, followed by a write of 0x0054, actually disables the watchdog timer. The watchdog timer will be re-enabled by any other write to this register.
	other	Normal clocking (32 kHz oscillator) for the watchdog timer. This is the condition after reset.

Secondary Watchdog Timer Register (SWDTR) (Address = 0x000C)		
Bit(s)	**Value**	**Description**
7:0		The time constant for the secondary watchdog timer is stored. This time constant will take effect the next time that the secondary watchdog counter counts down to zero. The timer counts modulo $n + 1$, where n is the programmed time constant. The secondary watchdog timer can be disabled by writing the sequence 0x005A – 0x0052 – 0x0044 to this register.

Global Output Control Register (GOCR) (Address = 0x000E)		
Bit(s)	**Value**	**Description**
7:6	00	CLK pin is driven by perclk.
	01	CLK pin is driven by perclk/2.
	10	CLK pin is low.
	11	CLK pin is high.
5:4	00	STATUS pin is active (low) during a first opcode byte fetch.
	01	STATUS pin is active (low) during an interrupt acknowledge.
	10	STATUS pin is low.
	11	STATUS pin is high.
3	1	WDTOUTB pin is low (1 cycle minimum, 2 cycles maximum, of 32 kHz).
	0	WDTOUTB pin follows watchdog function.
2	x	This bit is ignored.
1:0	00	/BUFEN pin is active (low) during external I/O cycles.
	01	/BUFEN pin is active (low) during data memory accesses.
	10	/BUFEN pin is low.
	11	/BUFEN pin is high.

Global ROM Configuration Register (GROM) (Address = 0x002C)		
Bit(s)	**Value**	**Description**
7 (Read-only)	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
4:0	00000	ROM identifier for this version of the chip.

Global RAM Configuration Register (GRAM) (Address = 0x002D)		
Bit(s)	**Value**	**Description**
7 (Read-only)	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
4:0	00001	RAM identifier for this version of the chip.

Global CPU Register (GCPU) (Address = 0x002E)		
Bit(s)	**Value**	**Description**
7 (read only)	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	read	These bits report the state of the SMODE pins.
4:0	00001	CPU identifier for all versions of the Rabbit 3000 chip.

Global Revision Register (GREV) (Address = 0x002F)		
Bit(s)	**Value**	**Description**
7 (read only)	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	read	These bits report the state of the SMODE pins.
4:0	00000	Revision identifier for Rabbit 3000 version of the chip.
	00001	Revision identifier for Rabbit 3000A version of the chip.

5. MEMORY MANAGEMENT

5.1 Overview

The Rabbit 3000 supports 8-bit external flash and SRAM devices; three chip selects and two read/write-enable strobes allow up to six external devices to be attached at once. The 8-bit mode allows 0, 1, 2, or 4 wait states to be specified for each device.

The Rabbit 3000's physical memory space contains four consecutive banks, each of which can be mapped to an individual chip-select/enable strobe pair. The banks can be set for equal sizes ranging from 128 KB up to 4 MB, providing a total physical memory range from 512 KB up to 16 MB. Figure 5-1 shows a sample configuration.

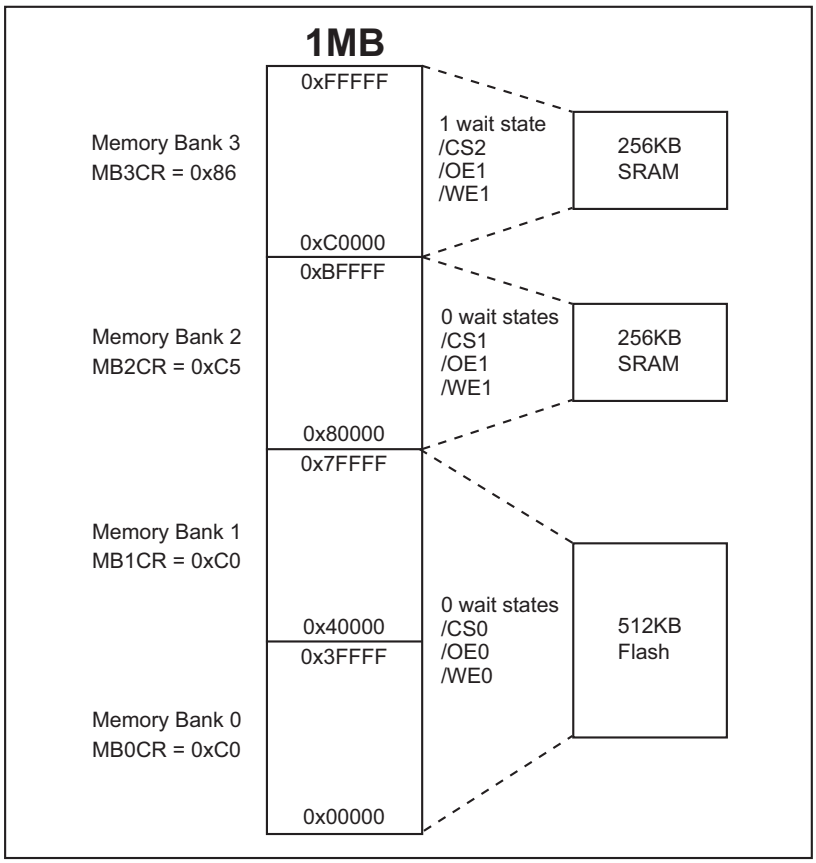


Figure 5-1. Mapping Rabbit 3000 Physical Memory Space

Either of the two most significant address bits (which are used to select the quadrant) can be inverted, providing the ability to bank-switch other pages from a larger memory device into the same memory bank.

Code is executed in the 64 KB logical memory space, which is divided into four segments: root, data, stack, and XPC. The root segment is mapped directly to physical address 0x00000, while the data and stack segments can be mapped to 4 KB boundaries anywhere in the physical space. The boundaries between the root and data segments and the data and stack segments can be adjusted in 4 KB blocks as well.

The XPC segment is a fixed 8 KB and points to a physical memory address specified in the XPC register. It is possible to run code in the XPC window, providing an easy means of storing and executing code beyond the 64 KB logical memory space. Special call and return instructions to physical addresses are provided that automatically update the XPC register as necessary.

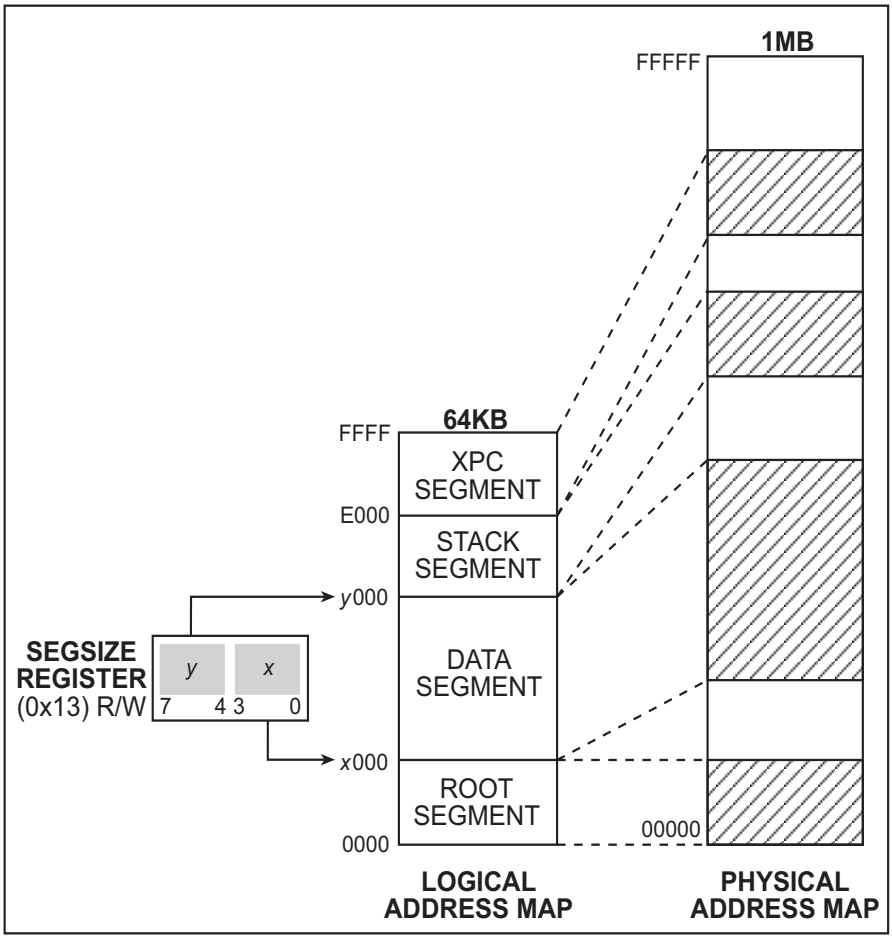


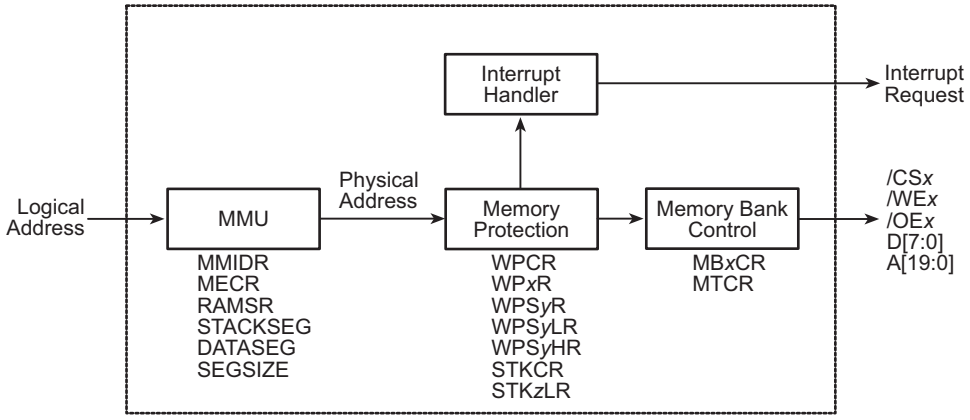
Figure 5-2. Logical and Physical Memory Mapping

The Rabbit 2000 and 3000 have numerous instructions for reading and writing data to logical addresses, but only limited support for reading and writing data to a physical memory address.

The 64 KB logical memory space limitation can also be expanded by using the separate instruction and data space mode. When this mode is enabled, address bit A16 is inverted for all data accesses in the root and/or data segments, while address bit A19 is inverted for all data accesses in the root and/or data segments *before* bank selection (physical device) occurs. These two features allow both code and data to access separate 64 KB logical spaces instead of sharing a single space.

It is possible to protect memory in the Rabbit 3000 at three different levels: each of the memory banks can be made read-only, physical memory can be write-protected in 64 KB blocks, and two of those 64 KB blocks can be protected with a granularity of 4 KB. A Priority 3 interrupt will occur if a write is attempted in one of the protected 64 KB or 4 KB blocks. In addition, it is possible to place limits around the code execution stack and generate an interrupt if a stack-related write occurs within 16 bytes of those limits.

5.1.1 Block Diagram



5.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
MMU Instruction/Data Register	MMIDR	0x0010	R/W	00000000	
Stack Segment Register	STACKSEG	0x0011	R/W	00000000	
Data Segment Register	DATASEG	0x0012	R/W	00000000	
Segment Size Register	SEGSIZE	0x0013	R/W	11111111	
Memory Bank 0 Control Register	MB0CR	0x0014	R/W	00001000	
Memory Bank 1 Control Register	MB1CR	0x0015	R/W	xxxxxxxx	
Memory Bank 2 Control Register	MB2CR	0x0016	R/W	xxxxxxxx	
Memory Bank 3 Control Register	MB3CR	0x0017	R/W	xxxxxxxx	
MMU Expanded Code Register	MECR	0x0018	R/W	xxxxx000	00000000
Memory Timing Control Register	MTCR	0x0019	W	xxxx0000	00000000
Write-Protect Control Register	WPCR	0x0440	W	—	00000000
Stack Limit Control Register	STKCR	0x0444	W	—	00000000
Stack Low Limit Register	STKLLR	0x0445	W	—	xxxxxxxx
Stack High Limit Register	STKHLR	0x0446	W	—	xxxxxxxx
RAM Segment Register	RAMSR	0x0448	W	—	00000000
Write Protect Low Register	WPLR	0x0460	W	—	00000000
Write Protect High Register	WPHR	0x0461	W	—	00000000
Write Protect Segment A Register	WPSAR	0x0480	W	—	00000000
Write Protect Segment A Low Register	WPSALR	0x0481	W	—	00000000
Write Protect Segment A High Register	WPSAHR	0x0482	W	—	00000000
Write Protect Segment B Register	WPSBR	0x0484	W	—	00000000
Write Protect Segment B Low Register	WPSBLR	0x0485	W	—	00000000
Write Protect Segment B High Register	WPSBHR	0x0486	W	—	00000000

5.2 Dependencies

5.2.1 I/O Pins

There are three chip select pins: /CS0, /CS1, and /CS2; two read strobes, /OE0 and /OE1; and two write strobes, /WE0 and /WE1.

There are eight dedicated data bus pins, D0 through D7.

There are 20 dedicated address pins, A0 through A19.

5.2.2 Clocks

All memory operations are clocked by the processor clock.

5.2.3 Interrupts

When a write is attempted to a write-protected 64 KB or 4 KB block, a write-protection violation interrupt is generated. The interrupt request is cleared when it is handled. The write-protection violation interrupt vector is in the IIR at offset 0x090. It is always set to Priority 3.

When a stack-related write is attempted to a region outside that set by the stack limit registers, a stack limit violation occurs. The interrupt request is cleared when it is handled. The stack limit violation interrupt vector is in the IIR at offset 0x1B0. It is always set to Priority 3.

5.3 Operation

5.3.1 Memory Management Unit (MMU)

Code execution takes place in the 64 KB logical memory space, which is divided into four segments: root, data, stack, and extended (XMEM). The root segment is always mapped starting at physical address 0x00000, but the other segments can be remapped to start at any physical 4 KB block boundary.

The data and stack segment mappings are set by writing to the appropriate register, as shown in Table 5-1. The DATASEG and STACKSEG registers provide backwards compatibility to the Rabbit 2000 processor.

Table 5-1. Memory Management Registers

Register	Segment	Size	Comments
DATASEG	Data	8 bits	—
STACKSEG	Stack	8 bits	—
XPC	XMEM	8 bits	Loaded via instructions **`LD XPC,A`** and **`LD A,XPC`**

Each of these registers provides a 4 KB offset that is added to the logical address to provide a physical address as shown in Figure 5-3.

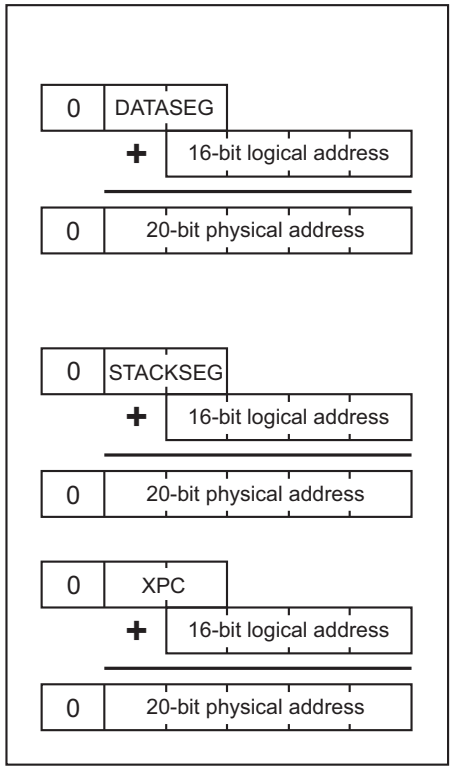


Figure 5-3. MMU Operation

5.3.2 8-bit Operation

On startup Memory Bank 0 is enabled to use /CS0, /OE0, and /WE0 with four wait states and write protection enabled; it is expected that an external flash device containing startup code be attached to those strobes. The other memory banks come up undefined and should be set via the appropriate MBxCR register to a valid setting before use.

The size of the memory banks can be defined in the MECR register. The default size is 256 KB (the bank selection looks at address bits 18 and 19), but this value can be adjusted down to 128 KB or up to 4 MB per bank.

The two address bits used to select the bank can be inverted in MBxCR, which enables mapping different sections of a memory device larger than the current memory bank into memory. An example of this feature is shown in Figure 5-4.

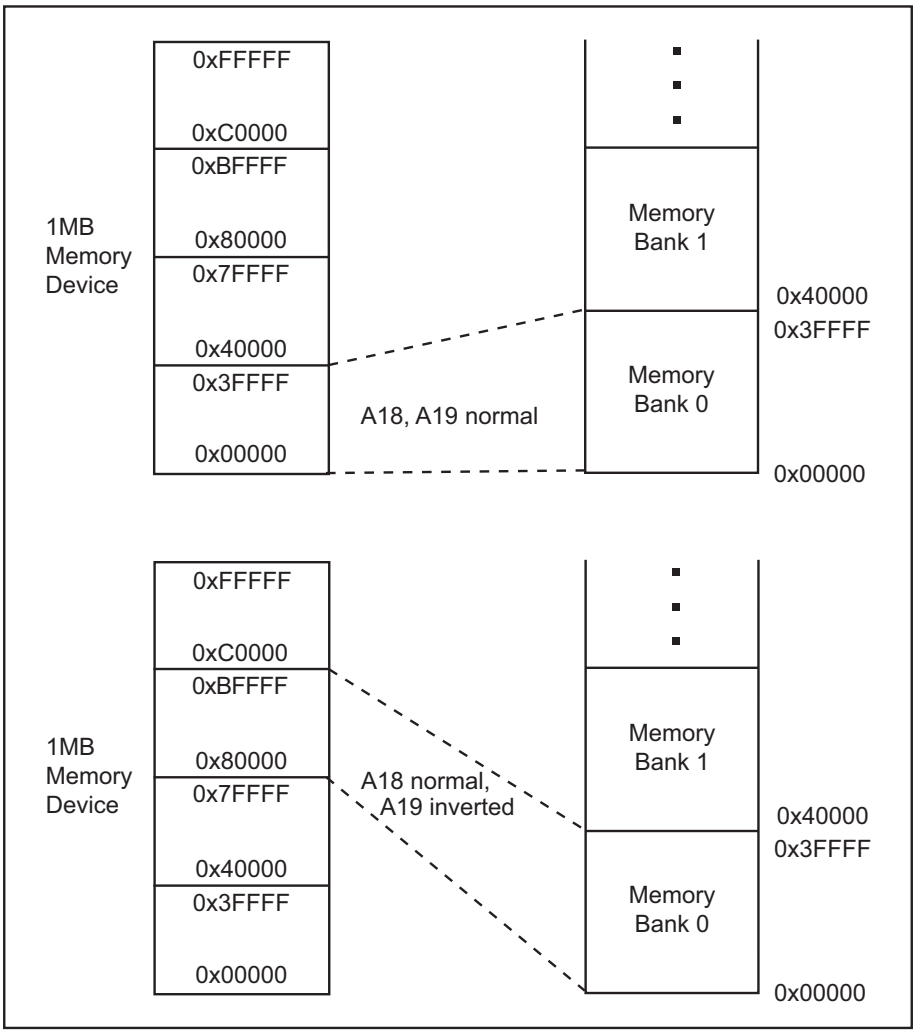


Figure 5-4. Mapping Different Sections of a Memory Device Larger Than the Current Memory Bank

It is possible to extend the timing of the /OE and/or /WE strobes by one half of a clock. This provides slightly longer strobes for slower memories; see the timing diagrams in Chapter 27. These options are available in MTCR.

It is possible to force /CS1 to be always active in MMIDR; enabling this will cause conflicts only if a device shares a /OE or /WE strobe with another device. This option allows faster access to particular memory devices.

5.3.3 Separate Instruction and Data Space

To make better use of the 64 KB of logical space, an option is provided to map code and data accesses in the same address space to separate devices. This is accomplished by enabling the inversion of A16 and the most-significant bit of the bank select bits for accesses in the root and data segments. Careful use of these features allows both code and data to separately use up to 64 KB of logical memory.

Starting with the Rabbit 3000A, the RAM segment register (RAMSR) provides a shortcut for updating code by accessing it as data. It provides a "window" that uses the instruction address decoding when read or written as data. This mapping will only occur when the RAMSR is within the root or data segements; the RAMSR will be ignored if it is mapped to the stack segement or XPC window.

The ***Rabbit 3000 Designer's Handbook*** provides further details on the use of the separate instruction and data space feature.

5.3.4 Memory Protection (Rabbit 3000A)

The ability to inhibit writes to physical memory was added starting with the Rabbit 3000A. The sixteen 64 KB physical memory blocks can be individually protected, and two of those blocks can additionally be subdivided and protected at a granularity of 4 KB. When a write is attempted, a new Priority 3 write-protection interrupt request is generated.

The write-protection can be enabled for the User mode only or for all modes.

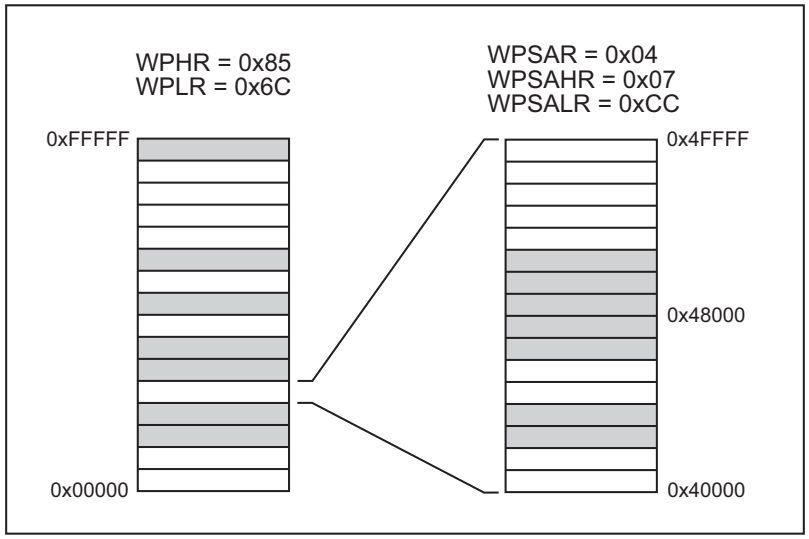


Figure 5-5. Sample Memory Protection Layout

5.3.5 Stack Protection (Rabbit 3000A)

The ability to detect stack overflow and underflow was added starting with the Rabbit 3000A. Low and high stack limits can be set on 256-byte boundaries. When a stack-relative memory access occurs within 16 bytes of these limits (or outside of them), a new Priority 3 stack violation interrupt occurs. The 16-byte buffer exists to allow stack protection even if the stack is placed against a memory segment boundary.

Figure 5-6 shows one possible stack layout. A 2048-byte stack is set up by setting STKHLR to 0x00E0, STKLLR to 0x00D8, and SP to 0xDFF0. Any stack-relative memory accesses above 0xDFEF (i.e., stack underflow) or below 0xD810 (i.e., overflow) would trigger the stack violation interrupt.

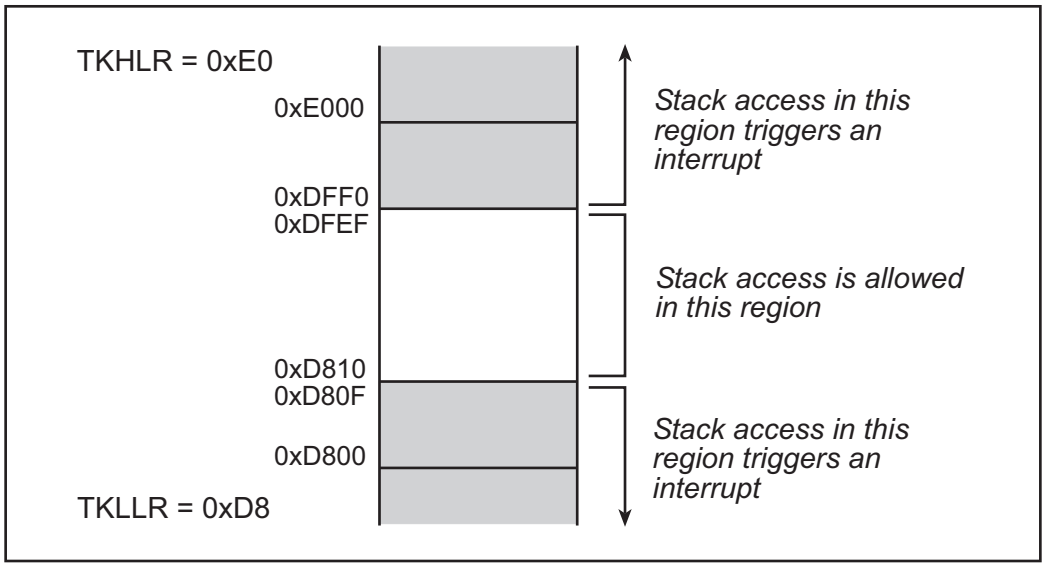


Figure 5-6. Simple Stack Protection Layout

5.3.6 RAM Segment Relocation (Rabbit 3000A)

Normally when instruction/data separation is enabled, instructions are stored in flash memory and data are stored in RAM memory. This can present a problem for the Interrupt Service Routine area, which often requires run-time modification. The RAM Segment Register (RAMSR) allows a 1, 2, or 4 KB segment of the logical memory space to be mapped as data would be mapped, even for program execution.

5.4 Register Descriptions

MMU Instruction/Data Register (MMIDR) (Address = 0x0010)		
Bit(s)	**Value**	**Description**
7*	0	Internal I/O addresses are decoded using only the lower eight bits of the internal I/O address bus. This restricts internal I/O addresses to the range 0x0000–0x00FF.
	1	Internal I/O addresses are decoded using all 15 bits of the address internal I/O address bus. This option must be selected to access internal I/O addresses of 0x0100 and higher.
6		This bit is reserved and must be written with zero.
5	0	Enable A16 and A19 inversion independent of instruction/data.
	1	Enable A16 and A19 inversion (controlled by bits 0-3) for data accesses only. This enables the instruction/data split for the separate I and D space.
4	0	Normal /CS1 operation.
	1	Force /CS1 always active. This will not cause any conflicts as long as the memory using /CS1 does not also share an Output Enable or Write Enable with another memory.
3	0	Normal operation.
	1	For a DATASEG access, invert A19 before MBxCR (bank select) decision.
2	0	Normal operation.
	1	For a data segment access: invert A16
1	0	Normal operation.
	1	For root access, invert A19 before MBxCR (bank select) decision.
0	0	Normal operation.
	1	For a root segment access: invert A16

* This bit is used only on the Rabbit 3000A and later versions; it is reserved and must be written with zero on the Rabbit 3000.

Stack Segment Register (STACKSEG) (Address = 0x0011)		
Bit(s)	**Value**	**Description**
7:0	Read	The current contents of this register are reported.
	Write	Eight LSBs (MSBs are set to zero by write) of physical address offset to use if SEGSIZ[7:4] ≤ Addr[15:12] < 0xE

Data Segment Register (DATASEG) (Address = 0x0012)		
Bit(s)	**Value**	**Description**
7:0	Read	The current contents of this register are reported.
	Write	Eight LSBs (MSBs are set to zero by write) of physical address offset to use if: SEGSIZ[3:0] ≤ Addr[15:12] < SEGSIZ[7:4]

Segment Size Register (SEGSIZ) (Address = 0x0013)		
Bit(s)	**Value**	**Description**
7:0	Read	The current contents of this register are reported.
7:4	Write	Boundary value for switching from DATASEG to STACKSEG for translation.
3:0	Write	Boundary value for switching from none to DATASEG for translation.

Memory Bank x Control Register (MB0CR) (Address = 0x0014) (MB1CR) (Address = 0x0015) (MB2CR) (Address = 0x0016) (MB3CR) (Address = 0x0017)		
Bit(s)	**Value**	**Description**
7:6	00	Four (five for writes) wait states for accesses in this bank.
	01	Two (three for writes) wait states for accesses in this bank.
	10	One (two for writes) wait states for accesses in this bank.
	11	Zero (one for writes) wait states for accesses in this bank.
5	0	Pass bank select address MSB for accesses in this bank.
	1	Invert bank select address MSB for accesses in this bank.
4	0	Pass bank select address LSB for accesses in this bank.
	1	Invert bank select address LSB for accesses in this bank.
3:2	00	/OE0 and /WE0 are active for accesses in this bank.
	01	/OE1 and /WE1 are active for accesses in this bank.
	10	/OE0 only is active for accesses in this bank (i.e., read-only). Transactions are normal in every other way.
	11	/OE1 only is active for accesses in this bank (i.e., read-only). Transactions are normal in every other way.
1:0	00	/CS0 is active for accesses in this bank.
	01	/CS1 is active for accesses in this bank.
	10	/CS2 is active for accesses in this bank.
	11	This bit combination is reserved and should not be used.

MMU Expanded Code Register (MECR) (Address = 0x0018)		
Bit(s)	**Value**	**Description**
7:3		These bits are reserved and should be written with zeros. Read returns zeros.
2:0	000	Normal operation.
	001	This bit combination is reserved and should not be used.
	010	This bit combination is reserved and should not be used.
	011	This bit combination is reserved and should not be used.
	100	For an XPC access, use MB0CR independent of bank select address.
	101	For an XPC access, use MB1CR independent of bank select address.
	110	For an XPC access, use MB2CR independent of bank select address.
	111	For an XPC access, use MB3CR independent of bank select address.

Memory Timing Control Register (MTCR) (Address = 0x0019)		
Bit(s)	**Value**	**Description**
7:4		These bits are reserved and should be written with zeros.
3	0	Normal timing for /OE1 (rising edge to rising edge, one clock minimum).
	1	Extended timing for /OE1 (one-half clock earlier than normal).
2	0	Normal timing for /OE0 (rising edge to rising edge, one clock minimum).
	1	Extended timing for /OE0 (one-half clock earlier than normal).
1	0	Normal timing for /WE1 (rising edge to falling edge, one and one-half clocks minimum).
	1	Extended timing for /WE1 (falling edge to falling edge, two clocks minimum).
0	0	Normal timing for /WE0 (rising edge to falling edge, one and one-half clocks minimum).
	1	Extended timing for /WE0 (falling edge to falling edge, two clocks minimum).

Write Protection Control Register (WPCR) (Address = 0x0440)		
Bit(s)	**Value**	**Description**
7:1		These bits are reserved and should be written with zeros.
0	0	Write protection in User Mode only.
	1	Write protection in System and User modes.

Stack Limit Control Register (STKCR) (Address = 0x0444)		
Bit(s)	**Value**	**Description**
7:1		These bits are reserved and should be written with zeros.
0	0	Disable stack-limit checking.
	1	Enable stack-limit checking.

Stack Low Limit Register (STKLLR) (Address = 0x0445)		
Bit(s)	**Value**	**Description**
7:0		Lower limit for stack-limit checking. If a stack operation or stack-relative memory access is attempted at an address less than {STKLLR, 0x0010}, a stack-limit violation interrupt is generated.

Stack High Limit Register (STKHLR) (Address = 0x0446)		
Bit(s)	**Value**	**Description**
7:0		Upper limit for stack-limit checking. If a stack operation or stack-relative memory access is attempted at an address greater than {STKHLR, 0x00EF}, a stack-limit violation interrupt is generated.

RAM Segment Register (RAMSR) (Address = 0x0448)		
Bit(s)	**Value**	**Description**
7:2		Compare value for RAM segment limit checking.
1:0	00	Disable RAM segment limit checking.
	01	Select data-type MMU translation if PC[15:10] is equal to RAMSR[7:2].
	10	Select data-type MMU translation if PC[15:11] is equal to RAMSR[7:3].
	11	Select data-type MMU translation if PC[15:12] is equal to RAMSR[7:4].

Write-Protect Low Register (WPLR) (Address = 0x0460)		
Bit(s)	**Value**	**Description**
7	0	Disable 64K write-protect for relative address 0x70000–0x7FFFF.
	1	Enable 64K write-protect for relative address 0x70000–0x7FFFF.
6	0	Disable 64K write-protect for relative address 0x60000–0x6FFFF.
	1	Enable 64K write-protect for relative address 0x60000–0x6FFFF.
5	0	Disable 64K write-protect for relative address 0x50000–0x5FFFF.
	1	Enable 64K write-protect for relative address 0x50000–0x5FFFF.
4	0	Disable 64Kwrite-protect for relative address 0x40000–0x4FFFF.
	1	Enable 64K write-protect for relative address 0x40000–0x4FFFF.
3	0	Disable 64K write-protect for relative address 0x30000–0x3FFFF.
	1	Enable 64Kwrite-protect for relative address 0x30000–0x3FFFF.
2	0	Disable 64K write-protect for relative address 0x20000–0x2FFFF.
	1	Enable 64K write-protect for relative address 0x20000–0x2FFFF.
1	0	Disable 64K write-protect for relative address 0x10000–0x1FFFF.
	1	Enable 64K write-protect for relative address 0x10000–0x1FFFF.
0	0	Disable 64K write-protect for relative address 0x00000–0x0FFFF.
	1	Enable 64K write-protect for relative address 0x00000–0x0FFFF.

Write-Protect High Register (WPHR) (Address = 0x0461)		
Bit(s)	**Value**	**Description**
7	0	Disable 64K write-protect for relative address 0xF0000–0xFFFFF.
	1	Enable 64K write-protect for relative address 0xF0000–0xFFFFF.
6	0	Disable 64K write-protect for relative address 0xE0000–0xEFFFF.
	1	Enable 64K write-protect for relative address 0xE0000–0xEFFFF.
5	0	Disable 64K write-protect for relative address 0xD0000–0xDFFFF.
	1	Enable 64K write-protect for relative address 0xD0000–0xDFFFF.
4	0	Disable 64K write-protect for relative address 0xC0000–0xCFFFF.
	1	Enable 64K write-protect for relative address 0xC0000–0xCFFFF.
3	0	Disable 64K write-protect for relative address 0xB0000–0xBFFFF.
	1	Enable 64K write-protect for relative address 0xB0000–0xBFFFF.
2	0	Disable 64K write-protect for relative address 0xA0000–0xAFFFF.
	1	Enable 64K write-protect for relative address 0xA0000–0xAFFFF.
1	0	Disable 64K write-protect for relative address 0x90000–0x9FFFF.
	1	Enable 64K write-protect for relative address 0x90000–0x9FFFF.
0	0	Disable 64K write-protect for relative address 0x80000–0x8FFFF.
	1	Enable 64K write-protect for relative address 0x80000–0x8FFFF.

Write-Protect Segment x Register (WPSAR) (Address = 0x0480) (WPSBR) (Address = 0x0484)		
Bit(s)	**Value**	**Description**
7:4		These bits are reserved and should be written with all zeros.
3:0		When these four bits match bits [19:16] of the physical address, write-protect that 64K range in 4K increments using WPSxLR and WPSxHR.

Write-Protect Segment x Low Register (WPSALR) (Address = 0x0481) (WPSBLR) (Address = 0x0485)		
Bit(s)	**Value**	**Description**
7	0	Disable 4K write protect for physical address 0x7000–0x7FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x7000–0x7FFF in WP Segment x.
6	0	Disable 4K write protect for physical address 0x6000–0x6FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x6000–0x6FFF in WP Segment x.
5	0	Disable 4K write protect for physical address 0x5000–0x5FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x5000–0x5FFF in WP Segment x.
4	0	Disable 4K write protect for physical address 0x4000–0x4FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x4000–0x4FFF in WP Segment x.
3	0	Disable 4K write protect for physical address 0x3000–0x3FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x3000–0x3FFF in WP Segment x.
2	0	Disable 4K write protect for physical address 0x2000–0x2FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x2000–0x2FFF in WP Segment x.
1	0	Disable 4K write protect for physical address 0x1000–0x1FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x1000–0x1FFF in WP Segment x.
0	0	Disable 4K write protect for physical address 0x0000–0x0FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x0000–0x0FFF in WP Segment x.

Write-Protect Segment x High Register (WPSAHR) (Address = 0x0482) (WPSBHR) (Address = 0x0486)		
Bit(s)	**Value**	**Description**
7	0	Disable 4K write protect for physical address 0xF000–0xFFFF in WP Segment x.
	1	Enable 4K write protect for physical address 0xF000–0xFFFF in WP Segment x.
6	0	Disable 4K write protect for physical address 0xE000–0xEFFF in WP Segment x.
	1	Enable 4K write protect for physical address 0xE000–0xEFFF in WP Segment x.
5	0	Disable 4K write protect for physical address 0xD000–0xDFFF in WP Segment x.
	1	Enable 4K write protect for physical address 0xD000–0xDFFF in WP Segment x.
4	0	Disable 4K write protect for physical address 0xC000–0xCFFF in WP Segment x.
	1	Enable 4K write protect for physical address 0xC000–0xCFFF in WP Segment x.
3	0	Disable 4K write protect for physical address 0xB000–0xBFFF in WP Segment x.
	1	Enable 4K write protect for physical address 0xB000–0xBFFF in WP Segment x.
2	0	Disable 4K write protect for physical address 0xA000–0xAFFF in WP Segment x.
	1	Enable 4K write protect for physical address 0xA000–0xAFFF in WP Segment x.
1	0	Disable 4K write protect for physical address 0x9000–0x9FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x9000–0x9FFF in WP Segment x.
0	0	Disable 4K write protect for physical address 0x8000–0x8FFF in WP Segment x.
	1	Enable 4K write protect for physical address 0x8000–0x8FFF in WP Segment x.

6. INTERRUPTS

6.1 Overview

The Rabbit 3000 can operate at one of four priority levels, 0–3, with Priority 0 being the expected standard operating level. The current priority and up to three previous priority levels are kept in the processor's 8-bit IP register, where bits 0–1 contain the current priority. Every time an interrupt is handled or an IPSET instruction occurs, the value in the register is shifted left by two bits, and the new priority placed in bits 0–1. When an IPRES or IRET instruction occurs, the value in IP is shifted right by two bits (bits 0–1 are shifted into bits 6–7). On reset, the processor starts at Priority 3.

Most interrupts can be set to be Priority 1–3. A pending interrupt will be handled only if its interrupt priority is greater than the current processor priority. This means that even a Priority 3 interrupt can be blocked if the processor is currently at Priority 3. The System Mode Violation, Stack Limit Violation, Write Protection Violation, secondary watchdog, and breakpoint interrupts are always enabled at Priority 3. In addition, when the system/user mode is enabled and the processor is in the user mode, the processor will not actually enter Priority 3; any attempt to enter Priority 3 will actually be requested as Priority 2.

When an interrupt is handled, a call is executed to a fixed location in the interrupt vector tables; this operation requires 10 clocks, the minimum interrupt latency for the Rabbit 3000. There are two vector tables, the internal and the external interrupt vector tables, that can be located anywhere in logical memory by setting the processor's IIR and EIR registers. The IIR and EIR registers hold the upper byte of each table's address. For example, if IIR is loaded with 0x00C4, then the internal interrupt vector table will start at the logical memory address 0xC400.

The internal interrupt vector table occupies 512 bytes, and the external interrupt vector table is 256 bytes in size. Since the RST and SYSCALL vectors use all eight bits of the IIR for addressing, the lowermost bit of IIR should always be set to zero so to keep some vectors from inadvertently overlapping.

Each interrupt's vector begins on a 16-byte boundary inside the vector tables. It may be possible to fit a small routine into that space, but it is typical to place a call to a separate routine in that location.

Some Rabbit 3000 instructions are "chained atomic," which means that an interrupt cannot occur between that instruction and the following instruction. These instructions are useful for doing things like exiting interrupt handlers properly or updating semaphores.

6.2 Operation

To ensure proper operation, all interrupt handler routines should be written according to the following guidelines.

- Push all registers to be used by the routine onto the stack before use, and pop them off the stack before returning from the ISR.
- Keep the ISR as short and fast as possible.
- If the ISR will run for some time, lower the interrupt priority as soon as possible within the ISR to allow other interrupts to occur.
- A number of special rules apply to interrupts when operating in the system/user mode; please see the appropriate chapter for more details.

6.3 Interrupt Tables

Table 6-1 shows the structure of the internal interrupt vector table. The first column is the vector address offset within the table. The second column shows the vectors in the first 256 bytes of the table, and the third column shows the vectors in the second 256 bytes.

Table 6-1. Internal Interrupt Vector Table Structure

Offset	0x0000 + Offset	0x0100 + Offset
0x0000	Periodic Interrupt	—
0x0010	Secondary Watchdog	—
0x0020	RST 10	—
0x0030	RST 18	—
0x0040	RST20	—
0x0050	RST 28	—
0x0060	Syscall instruction	—
0x0070	RST 38	PWM
0x0080	Slave Port	Sys/User Mode Violation
0x0090	Write Protect Violation	Quadrature Decoder
0x00A0	Timer A	Input Capture
0x00B0	Timer B	Stack Limit Violation
0x00C0	Serial Port A	Serial Port E
0x00D0	Serial Port B	Serial Port F
0x00E0	Serial Port C	Network Port A
0x00F0	Serial Port D	Timer C

Table 6-2 shows the structure of the external interrupt vector table. Each interrupt vector falls on a 16-byte boundary inside the table.

Table 6-2. External Interrupt Vector Table Structure

Offset	0x0000+
0x0000	External Interrupt 0
0x0010	External Interrupt 1
0x0020	—
0x0030	—
0x0040	—
0x0050	—
0x0060	—
0x0070	—

There is a priority among interrupts if multiple requests are pending, as shown in Table 6-3. Interrupts marked as "cleared automatically" have their requests cleared when the interrupt is first handled.

Table 6-3. Interrupt Priorities

Priority	Interrupt Source	Action Required to Clear the Interrupt
Highest	External 1	Automatically cleared by the interrupt acknowledge.
	External 0	Automatically cleared by the interrupt acknowledge.
	Periodic (2 kHz)	Read the status from the GCSR.
	Quadrature Decoder	Read the status from the QDCSR.
	Timer B	Read the status from the TBSR.
	Timer A	Read the status from the TASR.
	Input Capture	Read the status from the ICCSR.
	Slave Port	Rd: Read the data from the SPD0R, SPD1R or SPD2R. Wr: Write data to the SPD0R, SPD1R, SPD2R or write a dummy byte to the SPSR.
	Serial Port E	Rx: Read the data from the SEDR or SEAR. Tx: Write data to the SEDR, SEAR, SELR or write a dummy byte to the SESR.
	Serial Port F	Rx: Read the data from the SFDR or SFAR. Tx: Write data to the SFDR, SFAR, SFLR or write a dummy byte to the SFSR.
	Serial Port A	Rx: Read the data from the SADR or SAAR. Tx: Write data to the SADR, SAAR, SALR or write a dummy byte to the SASR.
	Serial Port B	Rx: Read the data from the SBDR or SBAR. Tx: Write data to the SBDR, SBAR, SBLR or write a dummy byte to the SBSR.
	Serial Port C	Rx: Read the data from the SCDR or SCAR. Tx: Write data to the SCDR, SCAR, SCLR or write a dummy byte to the SCSR.
Lowest	Serial Port D	Rx: Read the data from the SDDR or SDAR Tx: Write date to the SDDR, SDAR, SDLR or write a dummy byte to the SDSR

In the case of the external interrupts the only action that will clear the interrupt request is for the interrupt to take place, which automatically clears the request. A special action must be taken in the interrupt service routine for the other interrupts.

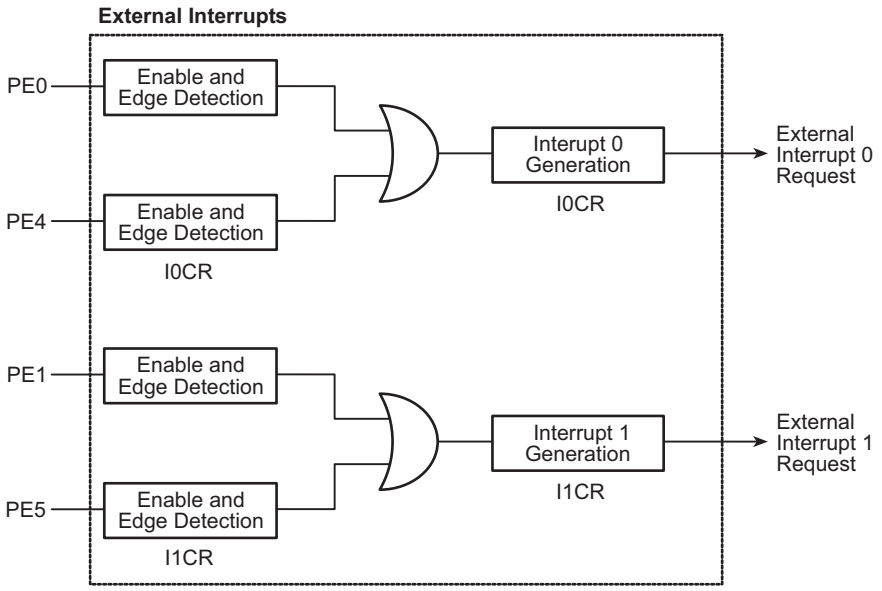
7. EXTERNAL INTERRUPTS

7.1 Overview

The Rabbit 3000 has two external interrupts. Each interrupt has two input pins that can be used to trigger the interrupt. The inputs have a pulse catcher that can detect rising, falling, or both edges. The pulse needs to be present for a least three peripheral clocks to be detected.

The signal on the external interrupt pin must be present for at least three peripheral clock cycles to be detected. In addition, the Rabbit 3000 has a minimum latency of 10 clocks to respond to an interrupt, so the minimum external interrupt response time is three peripheral clock cycles plus 10 processor clock cycles.

7.2 Block Diagram



7.2.1 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Interrupt 0 Control Register	I0CR	0x0098	W	xx000000
Interrupt 1 Control Register	I1CR	0x0099	W	xx000000

7.3 Dependencies

7.3.1 I/O Pins

The external interrupts can be enabled on pins PE0, PE1, PE4, and PE5. Each pin is associated with a particular interrupt vector as shown in Table 7-1 below.

Table 7-1. Rabbit 3000 Interrupt Vectors

Vector	Register	Pins
Interrupt 0	I0CR	PE0, PE4
Interrupt 1	I1CR	PE1, PE5

7.3.2 Clocks

The external interrupts are controlled by the peripheral clock. A pulse must be present for at least three peripheral clock cycles to trigger an interrupt.

7.3.3 Interrupts

An external interrupt is generated whenever the selected edge occurs on an enabled pin. The interrupt request is automatically cleared when the interrupt is handled.

The external interrupt vectors are in the EIR at offsets 0x000 and 0x010. They can be set as Priority 1, 2, or 3 in the appropriate I*x*CR.

7.4 Operation

The following steps must be taken to enable the external interrupts:

1. Write the vector(s) to the interrupt service routine to the external interrupt table.
2. Configure I*x*CR to select which pins are enabled for external interrupts, what edges are detected on each pin, and the interrupt priority.
3. When an interrupt occurs, read PEDR to determine which pin has a signal if more than one pin is enabled for a given external interrupt.

7.4.1 Example ISR

A sample interrupt handler is shown below.

```
extInt_isr::
; respond to external interrupt here
; interrupt is automatically cleared by interrupt acknowledge
ipres
ret
```

7.4.2 Expand Interrupts for Additional Peripheral Devices

When you need to expand the number of interrupts for additional peripheral devices, use the interrupt routine to dispatch interrupts to other virtual interrupt routines. Each additional interrupting device will have to signal the processor that it is requesting an interrupt. A separate signal line is needed for each device so that the processor can determine which devices are requesting an interrupt.

The following code shows how the interrupt service routines can be written.

```
; External interrupt Routine #0 (programmed priority could be 3)
int2:
    PUSH IP  ; save interrupt priority
    IPSET 1  ; set to priority really desired (1, 2, etc.)
; insert body of interrupt routine here
;
    POP IP   ; get back entry priority
    IPRES    ; restore interrupted routine's priority
    RET      ; return from interrupt
```

7.5 Register Descriptions

Interrupt x Control Register		(I0CR) (Address = 0x0098) (I1CR) (Address = 0x0099)
Bit(s)	**Value**	**Description**
7:6	xx	These bits are reserved for future use.
5:4	00	Parallel Port E high nibble interrupt disabled.
	01	Parallel Port E high nibble interrupt on falling edge.
	10	Parallel Port E high nibble interrupt on rising edge.
	11	Parallel Port E high nibble interrupt on both edges.
3:2	00	Parallel Port E low nibble interrupt disabled.
	01	Parallel Port E low nibble interrupt on falling edge.
	10	Parallel Port E low nibble interrupt on rising edge.
	11	Parallel Port E low nibble interrupt on both edges.
1:0	00	This external interrupt is disabled.
	01	This external interrupt uses Interrupt Priority 1.
	10	This external interrupt uses Interrupt Priority 2.
	11	This external interrupt uses Interrupt Priority 3.

8. PARALLEL PORT A

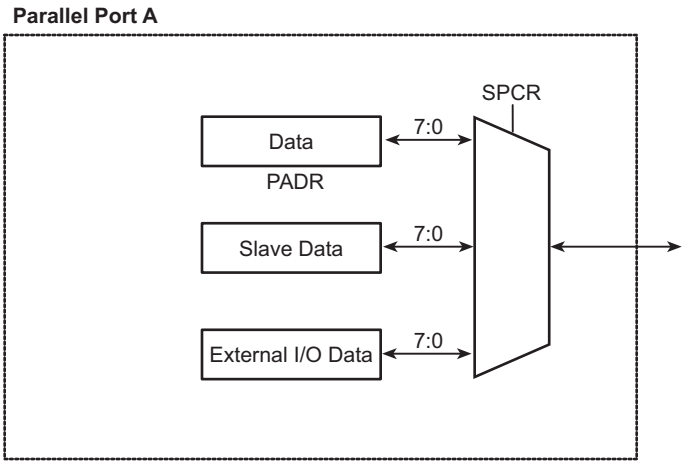
8.1 Overview

Parallel Port A is a byte-wide port that can be used as an input or an output port. Parallel Port A is also used as the data bus for the slave port and external I/O bus. The Slave Port Control Register (SPCR) is used to configure how Parallel Port A is used. Parallel Port A is an input at startup or reset. If the SMODE pins have selected the slave port bootstrap mode, Parallel Port A will be the slave port data bus until disabled by the processor. Parallel Port A can also be used as an external I/O data bus to isolate external I/O from the main data bus.

Table 8-1. Parallel Port A Pin Alternate Output Functions

Pin Name	Slave Port Data Bus	External I/O Bus
PA[7:0]	SD[7:0]	ID[7:0]

8.1.1 Block Diagram



8.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port A Data Register	PADR	0x0030	R/W	xxxxxxxx

8.2 Dependencies

8.2.1 I/O Pins

Parallel Port A uses pins PA0 through PA7. These pins can be used as follows.

- General-purpose 8-bit data input (write 0x080 to SPCR)
- General-purpose 8-bit data output (write 0x084 to SPCR)
- Slave port data bus (write 0x088 to SPCR)
- Data bus of the external I/O bus (write 0x08C to SPCR)

All Parallel Port A bits are inputs at startup or reset.

See the associated peripheral chapters for details on how they use Parallel Port A.

8.2.2 Clocks

Any outputs on Parallel Port A are clocked by the peripheral clock.

8.2.3 Other Registers

Register	Function
SPCR	Used to set up Parallel Port A.

8.2.4 Interrupts

There are no interrupts associated with Parallel Port A.

8.3 Operation

The following steps explain how to set up Parallel Port A.

1. Select the desired mode using SPCR.
2. If the slave port or external I/O bus is selected, refer to the chapters for those peripherals for further setup.

Once Parallel Port A is set up, data can be read or written by accessing PADR. Note that Parallel Port A is not available for general-purpose I/O while the slave port or the external I/O bus is selected. Selecting these options for Parallel Port A affects Parallel Port B because Parallel Port B is then used for address and control signals.

> **NOTE:** There may be a conflict in using Parallel Port A and Parallel Port F on the original Rabbit 3000 chip. Either Parallel Port A can be used as inputs, in which case Parallel Port F has full function, or if Parallel Port A cannot be used as inputs, use any pins on Parallel Port F not used for PWM or serial clock outputs as inputs and take the precaution of setting up Parallel Port F before the conflicting functionality of Parallel Port A is enabled. Refer to Rabbit's Technical Note TN228, ***Rabbit 3000 Parallel Port F Bug***, for more information.

8.4 Register Descriptions

Parallel Port A Data Register (PADR) (Address = 0x0030)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port A pins PA7–PA0 is reported.
	Write	The Parallel Port A buffer is written with this value for transfer to the Parallel Port A output register on the next rising edge of the peripheral clock.

Slave Port Control Register (SPCR) (Address = 0x0024)		
Bit(s)	**Value**	**Description**
7	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
	Write	These bits are ignored and should be written with zero.
4	0	/SCS from PE7.
	1	/SCS from PB6*.
3:2 (Write only)	00	Disable the slave port. Parallel Port A is a byte-wide input port.
	01	Disable the slave port. Parallel Port A is a byte-wide output port.
	10	Enable the slave port.
	11	Enable the external I/O bus. Parallel Port A is used for the data bus and Parallel Port B[7:2] is used for the address bus.
1:0 (Write only)	00	Slave port interrupts are disabled.
	01	Slave port interrupts use Interrupt Priority 1.
	10	Slave port interrupts use Interrupt Priority 2.
	11	Slave port interrupts use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

9. PARALLEL PORT B

9.1 Overview

Parallel Port B is a byte-wide port with each bit programmable for direction. The Parallel Port B pins are also used to access other peripherals on the chip—the slave port, the external I/O address bus, and clock I/O for clocked serial mode option for Serial Ports A and B. The Slave Port Control Register (SPCR) is used to configure how Parallel Port B is used when selecting the slave port or the external I/O bus modes.

When the slave port is enabled, either under program control or during parallel bootstrap, Parallel Port B pins carry the Slave Attention output signal, and four of the inputs carry the Slave Read strobe, Slave Write strobe, and Slave Address bits. The Slave Chip Select can also be programmed to come from a Parallel Port B pin.

When the external I/O bus option is enabled, six pins carry the external I/O address signals selected in SPCR.

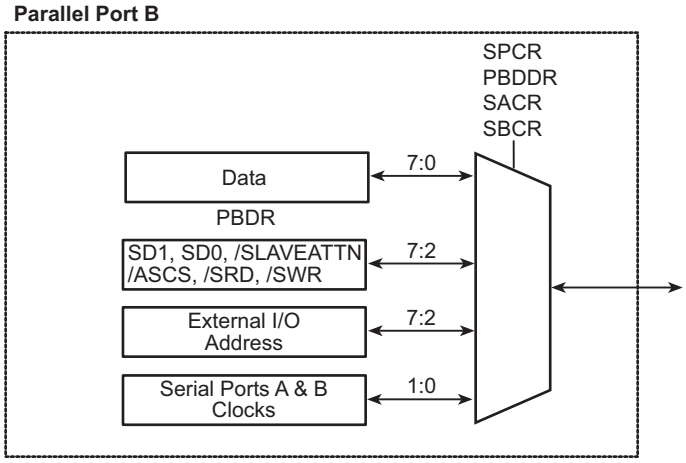
Two pins are used for the clocks for Serial Ports A and B when they are configured for the clocked serial mode. These two inputs can be used as clock outputs for these ports if selected in the respective serial port control registers. Note that the clocked serial output clock selection overrides all other programming for the two relevant Parallel Port B pins.

Table 9-1. Parallel Port B Pin Alternate Output Functions

Pin Name	Slave Port	Serial Ports A–D	External I/O Bus
PB7	/SLAVEATTN	—	IA5
PB6	/ASCS*	—	IA4
PB5	SA1	—	IA3
PB4	SA0	—	IA2
PB3	/SRD	—	IA1
PB2	/SWR	—	IA0
PB1	—	CLKA	—
PB0	—	CLKB	—

* Introduced with Rabbit 3000A chip

9.1.1 Block Diagram



9.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port B Data Register	PBDR	0x0040	R/W	00xxxxxx
Port B Data Direction Register	PBDDR	0x0047	W	11000000

9.2 Dependencies

9.2.1 I/O Pins

Parallel Port B uses pins PB0 through PB7. These pins can be used individually as data inputs or outputs; as the address bits for the external I/O bus; as control signals for the slave port; or as clocks for Serial Ports A and B.

On startup, bits 6 and 7 are outputs set low for backwards compatibility with the Rabbit 2000. All other pins are inputs.

Note that when the external I/O bus or slave port is enabled in SPCR, the Parallel Port B pins associated with those peripherals perform those actions, no matter what the settings are in PBDR or PBDDR. See the associated peripheral chapters for details on how they use Parallel Port B.

9.2.2 Clocks

All outputs on Parallel Port B are clocked by the peripheral clock (perclk).

9.2.3 Other Registers

Register	Function
SPCR	Sets the Parallel Port B function for some pins if the slave port or external I/O bus is enabled.

9.2.4 Interrupts

There are no interrupts associated with Parallel Port B.

9.3 Operation

The following steps must be taken before using Parallel Port B.

1. Select the desired input/output direction for each pin via PBDDR. Note that this setting is superseded for some pins if the slave port or external I/O bus is enabled in SPCR or if the clocked serial mode is enabled for Serial Ports A or B.
2. If the slave port or the external I/O bus is selected, refer to the chapters for those peripherals for further setup information.

Once the port is set up, data can be read or written by accessing PBDR. The value in PBDR of an output pin will reflect its current output value, but any value written to an input pin will not appear until that pin becomes an output.

9.4 Register Descriptions

Parallel Port B Data Register (PBDR) (Address = 0x0040)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port B pins PB7–PB0 is reported.
	Write	The Parallel Port B buffer is written with this value for transfer to the Parallel Port B output register on the next rising edge of the peripheral clock.

Parallel Port B Data Direction Register (PBDDR) (Address = 0x0047)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit is input.
	1	The corresponding port bit is an output.

Slave Port Control Register (SPCR) (Address = 0x0024)		
Bit(s)	**Value**	**Description**
7	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
	Write	These bits are ignored and should be written with zero.
4	0	/SCS from PE7.
	1	/SCS from PB6*.
3:2 (Write only)	00	Disable the slave port. Parallel Port A is a byte-wide input port.
	01	Disable the slave port. Parallel Port A is a byte-wide output port.
	10	Enable the slave port.
	11	Enable the external I/O bus. Parallel Port A is used for the data bus and Parallel Port B[7:2] is used for the address bus.
1:0 (Write only)	00	Slave port interrupts are disabled.
	01	Slave port interrupts use Interrupt Priority 1.
	10	Slave port interrupts use Interrupt Priority 2.
	11	Slave port interrupts use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

10. PARALLEL PORT C

10.1 Overview

Parallel Port C is a byte-wide port that has four inputs and four outputs. The even-numbered ports—PC0, PC2, PC4, and PC6—are outputs. The odd-numbered ports—PC1, PC3, PC5, and PC7—are inputs.. These are simple inputs and outputs controlled and reported in the Port C Data Register (PCDR).

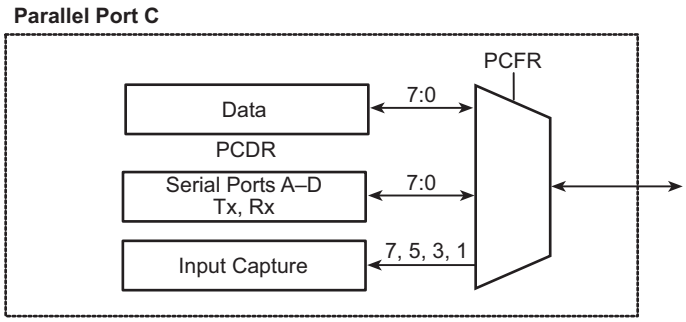
The Parallel Port C pins may be used as serial port pins, and the inputs can be used as inputs to the Input Capture peripheral.

Table 10-1. Parallel Port C Alternate Pin Functions

Pin Name	Outputs	Inputs	
	Serial Ports A–D	Serial Ports A–D	Input Capture
PC7	—	RXA	×
PC6	TXA	—	—
PC5	—	RXB	×
PC4	TXB	—	—
PC3	—	RXC	×
PC2	TXC	—	—
PC1	—	RXD	×
PC0	TXD	—	—

After reset, the default condition for Parallel Port C is four outputs (the even-numbered bits) and four inputs (the odd-numbered bits). For compatibility with the Rabbit 2000 microprocessor, these outputs are driven with a logic zero (low) on PC6 and a logic one (high) on PC4, PC2, and PC0. When PCDR is read, the logic level on the pin is returned. If the pin is an output, the value it is set to is returned.

10.1.1 Block Diagram



10.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port C Data Register	PCDR	0x0050	R/W	x0x1x1x1
Port C Function Register	PCFR	0x0055	W	x0x0x0x0

10.2 Dependencies

10.2.1 I/O Pins

Parallel Port C uses pins PC0 through PC7. These pins can be used individually as data inputs or outputs; as serial port transmit and receive for Serial ports A–D. The Input Capture peripheral can also watch pins PC7, PC5, PC3, and PC1.

On startup, PC4, PC2, and PC0 are outputs set high, PC6 is set low, and the other pins are inputs for compatibility with the Rabbit 2000.

See Chapter 20 for details on how the Input Capture peripheral uses Parallel Port C.

10.2.2 Clocks

All outputs on Parallel Port C are clocked by the peripheral clock.

10.2.3 Other Registers

Register	Function
SACR, SBCR, SCCR, SDCR	Select a Parallel Port C pin as serial data input.
ICS1R, ICS2R	Select a Parallel Port C pin as a start/stop condition for the Input Capture peripheral.

10.2.4 Interrupts

There are no interrupts associated with Parallel Port C.

10.3 Operation

When PCDR is read, bits 1, 3, 5, and 7 return the logic level on the pin. Bits 0, 2, 4, and 6 return the value of the signal driving the output buffers. The signal driving the output buffers and the value of the output pin are normally the same. The output pins are driven either by PCDR or by one of the serial port transmit lines. The bits set in the PCFR identify whether the data register or the serial port transmit lines were driving the pins. When a parallel port output pin is selected to be a serial port output, the value stored in PCDR is ignored.

The data line inputs can be set up as asynchronous or clocked serial inputs via bits 4, 5, and 6 of the corresponding Serial Port Control Registers (SxCR). When serving as serial inputs, the data lines can still be read from PCDR.

On reset the active (even-numbered) function register bits are zeroed, causing Parallel Port C to behave as an I/O port. Bit 6 of the Port C data register is zeroed while the remaining even numbered bits are set to 1.

10.4 Register Descriptions

Parallel Port C Data Register (PCDR) (Address = 0x0050)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port C pins PC7–PC0 is reported.
	Write	The Parallel Port C buffer is written with this value for transfer to the Parallel Port C output register on the next rising edge of the peripheral clock.

Parallel Port C Function Register (PCFR) (Address = 0x0055)		
Bit(s)	**Value**	**Description**
7	x	Parallel Port C pin 7 is always a parallel port input.
6	0	Parallel Port C pin 6 is a parallel port output.
	1	Parallel Port C pin 6 drives TxA.
5	x	Parallel Port C pin 5 is always a parallel port input.
4	0	Parallel Port C pin 4 is a parallel port output.
	1	Parallel Port C pin 4 drives TxB.
3	x	Parallel Port C pin 3 is always a parallel port input.
2	0	Parallel Port C pin 2 is a parallel port output.
	1	Parallel Port C pin 2 drives TxC.
1	x	Parallel Port C pin 1 is always a parallel port input.
0	0	Parallel Port C pin 0 is a parallel port output.
	1	Parallel Port C pin 0 drives TxD.

Serial Port x Control Register		(SACR) (Address = 0x00C4) (SBCR) (Address = 0x00D4)
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynch mode.
	01	In clocked serial mode, start a byte receive operation.
	10	In clocked serial mode, start a byte transmit operation.
	11	In clocked serial mode, start a byte transmit operation and a byte receive operation simultaneously.
5:4	00	Parallel Port C is used for input.
	01	Parallel Port D is used for input.
	1x	Disable the receiver input.
3:2	00	Asynch mode with 8 bits per character.
	01	Asynch mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	Clocked serial mode with external clock. Serial Port A clock is on Parallel Port B pin 1 Serial Port B clock is on Parallel Port B pin 0
	11	Clocked serial mode with internal clock. Serial Port A clock is on Parallel Port B pin 1 Serial Port B clock is on Parallel Port B pin 0
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.

Serial Port x Control Register (SCCR) (Address = 0x00E4) (SDCR) (Address = 0x00F4)		
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynch mode.
	01	In clocked serial mode, start a byte receive operation.
	10	In clocked serial mode, start a byte transmit operation.
	11	In clocked serial mode, start a byte transmit operation and a byte receive operation simultaneously.
5	0	Enable the receiver input.
	1	Disable the receiver input.
4	x	This bit is ignored.
3:2	00	Asynch mode with 8 bits per character.
	01	Asynch mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	Clocked serial mode with external clock. Serial Port C clock is on Parallel Port F pin 1 Serial Port D clock is on Parallel Port F pin 0
	11	Clocked serial mode with internal clock. Serial Port C clock is on Parallel Port F pin 1 Serial Port D clock is on Parallel Port F pin 0
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.
	11	The serial port uses Interrupt Priority 3.

11. PARALLEL PORT D

11.1 Overview

Parallel Port D is a byte-wide port with each bit programmable individually for data direction and drive level. These are simple inputs and outputs controlled and reported in the Port D Data Register (PDDR). The output registers are cascaded and timer-controlled, making it possible to generate precise timing pulses.

Four of the Parallel Port D pins have alternate functions as Serial Ports A and B. Alternate serial port bit assignments make it possible for the same serial port to connect to different communications lines that are not operating at the same time.

When used as outputs, the Parallel Port D bits are buffered, with the data written to PDDR transferred to the output pins on a selected timing edge. Either the peripheral clock or the outputs of Timer A1, Timer B1, or Timer B2 can be used for this function, with each nibble of the port having a separate select field to control this timing. Each bit can either be programmed as an open-drain output or a standard output.

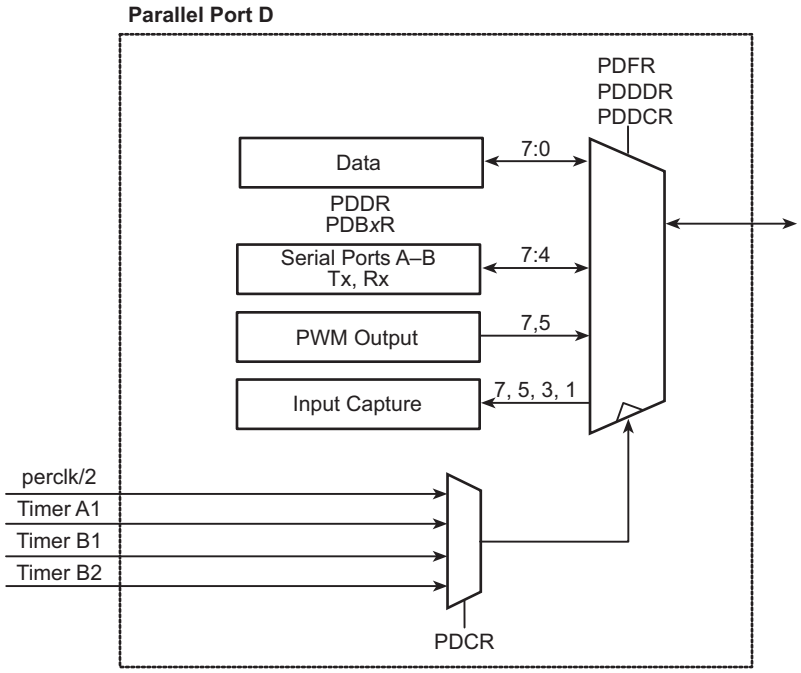
Because of the buffered nature of Parallel Port D, using a read-modify-write type of operation can lead to old data being written to PDDR. To alleviate this potential problem, each bit of the port can be written individually using a separate address for each bit.

Table 11-1. Parallel Port D Alternate Pin Functions

Pin Name	Inputs		Outputs	
	Serial Ports A–B	Input Capture	PWM	Serial Ports A–B
PD7	ARXA	×	APWM3*	—
PD6	—	—	—	ATXA
PD5	ARXB	×	APWM2*	—
PD4	—	—	—	ATXB
PD3	—	×	—	—
PD2	—	—	—	—
PD1	—	×	—	—
PD0	—	—	—	—

* Introduced with Rabbit 3000A chip.

11.1.1 Block Diagram



11.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port D Data Register	PDDR	0x0060	R/W	xxxxxxxx
Port D Control Register	PDCR	0x0064	W	xx00xx00
Port D Function Register	PDFR	0x0065	W	xxxxxxxx
Port D Drive Control Register	PDDCR	0x0066	W	xxxxxxxx
Port D Data Direction Register	PDDDR	0x0067	W	00000000
Port D Bit 0 Register	PDB0R	0x0068	W	xxxxxxxx
Port D Bit 1 Register	PDB1R	0x0069	W	xxxxxxxx
Port D Bit 2 Register	PDB2R	0x006A	W	xxxxxxxx
Port D Bit 3 Register	PDB3R	0x006B	W	xxxxxxxx
Port D Bit 4 Register	PDB4R	0x006C	W	xxxxxxxx
Port D Bit 5 Register	PDB5R	0x006D	W	xxxxxxxx
Port D Bit 6 Register	PDB6R	0x006E	W	xxxxxxxx
Port D Bit 7 Register	PDB7R	0x006F	W	xxxxxxxx

11.2 Dependencies

11.2.1 I/O Pins

Parallel Port D uses pins PD0 through PD7. These pins can be used individually as data inputs or outputs, as serial port transmit and receive for Serial Ports A and B, or as PWM outputs. The input capture peripheral can also watch pins PD7, PD5, PD3, and PD1..

All pins are set as inputs on startup.

The individual bits can be set to be open-drain outputs or standard outputs via PDDCR.

11.2.2 Clocks

All outputs on Parallel Port D are clocked by the peripheral clock divided by 2 unless changed in PDCR, where the option of updating the Parallel Port D pins can be synchronized to the output of Timer A1, Timer B1, or Timer B2.

11.2.3 Other Registers

Register	Function
SACR, SBCR	Select a Parallel Port D pin as serial data input.

11.2.4 Interrupts

There are no interrupts associated with Parallel Port D.

11.3 Operation

The following steps must be taken before using Parallel Port D.

1. Select the desired input/output direction for each pin via PDDDR.
2. Select standard high/low or open-drain functionality for outputs via PDDCR.
3. If you plan to use a pin pair as a serial port, select it via PDFR (transmit/output) and set up the serial inputs (receive) as asynchronous or clocked via bits 4, 5, and 6 of the corresponding Serial Port Control Registers (SxCR).

Once Parallel Port D is set up, data can be read or written by accessing PDDR. The value of an output pin read in from PDDR will reflect its current output value, but any value written to an input pin will not appear until that pin becomes an output.

On reset, PDDDR is zeroed, making all pins inputs. In addition bits 0, 1, 4, and 5 in PDCR are zeroed to ensure that data are clocked into the output registers when loaded. All other registers associated with Parallel Port D are not initialized on reset.

11.4 Register Descriptions

Parallel Port D Data Register (PDDR) (Address = 0x0060)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port D pins PD7–PD0 is reported.
	Write	The Parallel Port D buffer is written with this value for transfer to the Parallel Port D output register on the next rising edge of the peripheral clock.

Parallel Port D Control Register (PDCR) (Address = 0x0064)		
Bit(s)	**Value**	**Description**
7:6	00	These bits are ignored and should be written with zero.
5:4	00	The upper nibble peripheral clock is perclk/2.
	01	The upper nibble peripheral clock is the output of Timer A1.
	10	The upper nibble peripheral clock is the output of Timer B1.
	11	The upper nibble peripheral clock is the output of Timer B2.
3:2	00	These bits are ignored and should be written with zero.
1:0	00	The lower nibble peripheral clock is perclk/2.
	01	The lower nibble peripheral clock is the output of Timer A1.
	10	The lower nibble peripheral clock is the output of Timer B1.
	11	The lower nibble peripheral clock is the output of Timer B2.

Parallel Port D Function Register (PDFR) (Address = 0x0065)		
Bit(s)	**Value**	**Description**
7	x	Parallel Port D pin 7 is always a parallel port I/O pin.
6	0	Parallel Port D pin 6 is a parallel port I/O pin.
	1	Parallel Port D pin 6 drives ATxA.
5	x	Parallel Port D pin 5 is always a parallel port I/O pin.
4	0	Parallel Port D pin 4 is a parallel port I/O pin.
	1	Parallel Port D pin 4 drives ATxB.
3:0	x	Parallel Port D pins 3:0 are always parallel port I/O pins.

Parallel Port D Drive Control Register (PDDCR) (Address = 0x0066)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit, as an output, is driven high and low.
	1	The corresponding port bit, as an output, is open-drain.

Parallel Port D Data Direction Register (PDDDR) (Address = 0x0067)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit is an input.
	1	The corresponding port bit is an output.

Parallel Port D Bit 0 Register (PDB0R) (Address = 0x0068)		
Bit(s)	**Value**	**Description**
7:1		These bits are ignored.
0	Write	The port buffer (bit 0) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 1 Register (PDB1R) (Address = 0x0069)		
Bit(s)	**Value**	**Description**
7:2,0		These bits are ignored.
1	Write	The port buffer (bit 1) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 2 Register (PDB2R) (Address = 0x006A)		
Bit(s)	**Value**	**Description**
7:3,1:0		These bits are ignored.
2	Write	The port buffer (bit 2) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 3 Register (PDB3R) (Address = 0x006B)		
Bit(s)	**Value**	**Description**
7:4,2:0		These bits are ignored.
3	Write	The port buffer (bit 3) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 4 Register (PDB4R) (Address = 0x006C)		
Bit(s)	**Value**	**Description**
7:5,3:0		These bits are ignored.
4	Write	The port buffer (bit 4) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 5 Register (PDB5R) (Address = 0x006D)		
Bit(s)	**Value**	**Description**
7:6,4:0		These bits are ignored.
5	Write	The port buffer (bit 5) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 6 Register (PDB6R) (Address = 0x006E)		
Bit(s)	**Value**	**Description**
7,5:0		These bits are ignored.
6	Write	The port buffer (bit 6) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port D Bit 7 Register (PDB7R) (Address = 0x006F)		
Bit(s)	**Value**	**Description**
6:0		These bits are ignored.
7	Write	The port buffer (bit 7) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Serial Port x Control Register		(SACR) (Address = 0x00C4) (SBCR) (Address = 0x00D4)
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynch mode.
	01	In clocked serial mode, start a byte receive operation.
	10	In clocked serial mode, start a byte transmit operation.
	11	In clocked serial mode, start a byte transmit operation and a byte receive operation simultaneously.
5:4	00	Parallel Port C is used for input.
	01	Parallel Port D is used for input.
	1x	Disable the receiver input.
3:2	00	Asynch mode with 8 bits per character.
	01	Asynch mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	Clocked serial mode with external clock. Serial Port A clock is on Parallel Port B pin 1 Serial Port B clock is on Parallel Port B pin 0
	11	Clocked serial mode with internal clock. Serial Port A clock is on Parallel Port B pin 1 Serial Port B clock is on Parallel Port B pin 0
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.

12. PARALLEL PORT E

12.1 Overview

Parallel Port E is a byte-wide port with each bit programmable for data direction. These are simple inputs and outputs controlled and reported in the Port E Data Register (PEDR).

Each of the Parallel Port E outputs can be configured as an I/O strobe. In addition, four of the Parallel Port E lines can be used as interrupt request inputs.

When used as outputs, the Parallel Port E bits are buffered, with the data written to PEDR transferred to the output pins on a selected timing edge. Either the peripheral clock or the outputs of Timer A1, Timer B1, or Timer B2 can be used for this function, with each nibble of the port having a separate select field to control this timing. Each bit can either be programmed as open-drain or driven high and low.

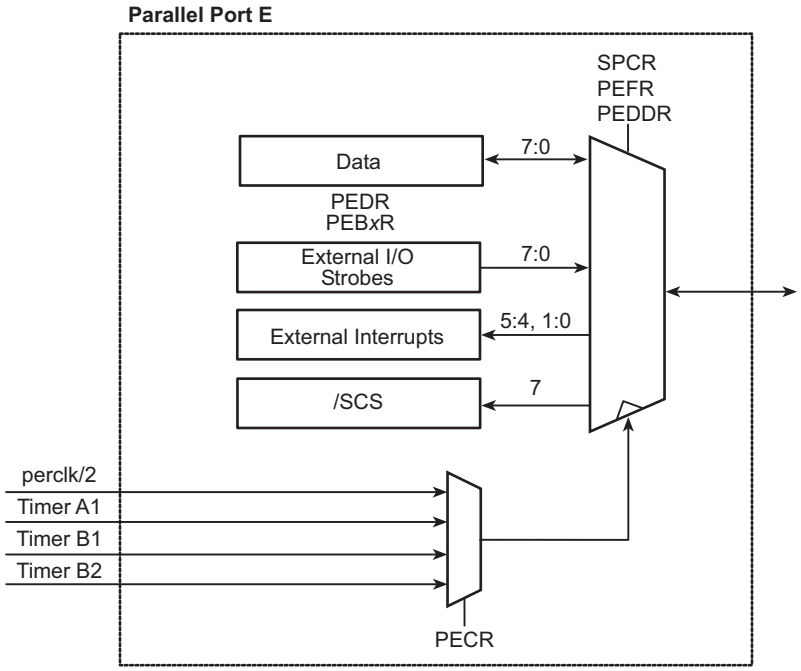
Because of the buffered nature of Parallel Port E, using a read-modify-write type of operation can lead to old data being written to PEDR. To alleviate this potential problem, each bit of the port can be written individually using a separate address for each bit.

Bit 7 of Parallel Port E is used as the default chip select input for the slave port when the slave port is enabled, either for parallel bootstrap or under program control.

Table 12-1. Parallel Port E Alternate Pin Functions

Pin Name	Inputs		Outputs
	Interrupts	Slave Port	I/O Strobes
PE7	—	/SCS	I7
PE6	—		I6
PE5	INT1B		I5
PE4	INT0B		I4
PE3	—		I3
PE2	—		I2
PE1	INT1A		I1
PE0	INT0A		I0

12.1.1 Block Diagram



12.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port E Data Register	PEDR	0x0070	R/W	xxxxxxxx
Port E Control Register	PECR	0x0074	W	xx00xx00
Port E Function Register	PEFR	0x0075	W	00000000
Port E Data Direction Register	PEDDR	0x0077	W	00000000
Port E Bit 0 Register	PEB0R	0x0078	W	xxxxxxxx
Port E Bit 1 Register	PEB1R	0x0079	W	xxxxxxxx
Port E Bit 2 Register	PEB2R	0x007A	W	xxxxxxxx
Port E Bit 3 Register	PEB3R	0x007B	W	xxxxxxxx
Port E Bit 4 Register	PEB4R	0x007C	W	xxxxxxxx
Port E Bit 5 Register	PEB5R	0x007D	W	xxxxxxxx
Port E Bit 6 Register	PEB6R	0x007E	W	xxxxxxxx
Port E Bit 7 Register	PEB7R	0x007F	W	xxxxxxxx

12.2 Dependencies

12.2.1 I/O Pins

Parallel Port E uses the pins PE0 through PE7. These pins can be used individually as data inputs or outputs, or as external I/O strobes; four of the Parallel Port E lines can be used as interrupt request inputs. There is also an option to input the slave port chip select on PE7.

All pins are set as inputs on startup, and the alternate pin functions are disabled.

12.2.2 Clocks

All outputs on Parallel Port E are clocked by the peripheral clock divided by two unless changed in PECR, where the option of updating the Parallel Port E pins can be synchronized to the output of Timer A1, Timer B1, or Timer B2.

12.2.3 Other Registers

Register	Function
I0CR, I1CR	Select a Parallel Port E pin as an external interrupt input.
SPCR	Select slave chip select on PE7.

12.2.4 Interrupts

External interrupts can be accepted from pins PE5, PE4, PE1 or PE0; see Chapter 7 for more details.

12.3 Operation

The following steps must be taken before using Parallel Port E.

1. Select the desired input/output direction for each pin via PEDDR.
2. Use PECR to select the clock.
3. If an alternative function is desired for a pin, select it via PEFR, I0CR, I1CR, or SPCR.

Once the port is set up, data can be read or written by accessing PEDR. The value of an output pin read in from PEDR will reflect its current output value, but any value written to an input pin will not appear until that pin becomes an output.

On reset, PEDDR and PEFR are zeroed, making all pins inputs, and disabling the alternate output functions. In addition certain bits 0, 1, 4, and 5 in PECR are zeroed to ensure that data are clocked into the output registers when loaded. All other registers associated with Parallel Port E are not initialized on reset.

12.4 Register Descriptions

Parallel Port E Data Register (PEDR) (Address = 0x0070)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port E pins PE7–PE0 is reported.
	Write	The Parallel Port E buffer is written with this value for transfer to the Parallel Port E output register on the next rising edge of the peripheral clock.

Parallel Port E Control Register (PECR) (Address = 0x0074)		
Bit(s)	**Value**	**Description**
7:6	00	These bits are ignored and should be written with zero.
5:4	00	The upper nibble peripheral clock is perclk/2.
	01	The upper nibble peripheral clock is the output of Timer A1.
	10	The upper nibble peripheral clock is the output of Timer B1.
	11	The upper nibble peripheral clock is the output of Timer B2.
3:2	00	These bits are ignored and should be written with zero.
1:0	00	The lower nibble peripheral clock is perclk/2.
	01	The lower nibble peripheral clock is the output of Timer A1.
	10	The lower nibble peripheral clock is the output of Timer B1.
	11	The lower nibble peripheral clock is the output of Timer B2.

Parallel Port E Function Register (PEFR) (Address = 0x0075)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit is a parallel port I/O pin.
	1	The corresponding port bit is an I/O strobe.

Parallel Port E Data Direction Register (PEDDR) (Address = 0x0077)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit is input.
	1	The corresponding port bit is an output.

Parallel Port E Bit 0 Register (PEB0R) (Address = 0x0078)		
Bit(s)	**Value**	**Description**
7:1		These bits are ignored.
0	Write	The port buffer (bit 0) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 1 Register (PEB1R) (Address = 0x0079)		
Bit(s)	**Value**	**Description**
7:2,0		These bits are ignored.
1	Write	The port buffer (bit 1) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 2 Register (PEB2R) (Address = 0x007A)		
Bit(s)	**Value**	**Description**
7:3,1:0		These bits are ignored.
2	Write	The port buffer (bit 2) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 3 Register (PEB3R) (Address = 0x007B)		
Bit(s)	**Value**	**Description**
7:4,2:0		These bits are ignored.
3	Write	The port buffer (bit 3) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 4 Register (PEB4R) (Address = 0x007C)		
Bit(s)	**Value**	**Description**
7:5,3:0		These bits are ignored.
4	Write	The port buffer (bit 4) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 5 Register (PEB5R) (Address = 0x007D)		
Bit(s)	**Value**	**Description**
7:6,4:0		These bits are ignored.
5	Write	The port buffer (bit 5) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 6 Register (PEB6R) (Address = 0x007E)		
Bit(s)	**Value**	**Description**
7,5:0		These bits are ignored.
6	Write	The port buffer (bit 6) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Parallel Port E Bit 7 Register (PEB7R) (Address = 0x007F)		
Bit(s)	**Value**	**Description**
6:0		These bits are ignored.
7	Write	The port buffer (bit 7) is written with the value of this bit. The port buffer will be transferred to the port output register on the next rising edge of the peripheral clock

Interrupt x Control Register		(I0CR) (Address = 0x0098) (I1CR) (Address = 0x0099)
Bit(s)	**Value**	**Description**
7:6	xx	These bits are reserved for future use.
5:4	00	Parallel Port E high nibble interrupt disabled.
	01	Parallel Port E high nibble interrupt on falling edge.
	10	Parallel Port E high nibble interrupt on rising edge.
	11	Parallel Port E high nibble interrupt on both edges.
3:2	00	Parallel Port E low nibble interrupt disabled.
	01	Parallel Port E low nibble interrupt on falling edge.
	10	Parallel Port E low nibble interrupt on rising edge.
	11	Parallel Port E low nibble interrupt on both edges.
1:0	00	This external interrupt is disabled.
	01	This external interrupt uses Interrupt Priority 1.
	10	This external interrupt uses Interrupt Priority 2.
	11	This external interrupt uses Interrupt Priority 3.

Slave Port Control Register (SPCR) (Address = 0x0024)		
Bit(s)	**Value**	**Description**
7	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
	Write	These bits are ignored and should be written with zero.
4	0	/SCS from PE7.
	1	/SCS from PB6*.
3:2 (Write only)	00	Disable the slave port. Parallel Port A is a byte-wide input port.
	01	Disable the slave port. Parallel Port A is a byte-wide output port.
	10	Enable the slave port.
	11	Enable the external I/O bus. Parallel Port A is used for the data bus and Parallel Port B[7:2] is used for the address bus.
1:0 (Write only)	00	Slave port interrupts are disabled.
	01	Slave port interrupts use Interrupt Priority 1.
	10	Slave port interrupts use Interrupt Priority 2.
	11	Slave port interrupts use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

13. PARALLEL PORT F

13.1 Overview

Parallel Port F is a byte-wide port with each bit programmable for data direction. These are simple inputs and outputs controlled and reported in the Port F Data Register (PFDR).

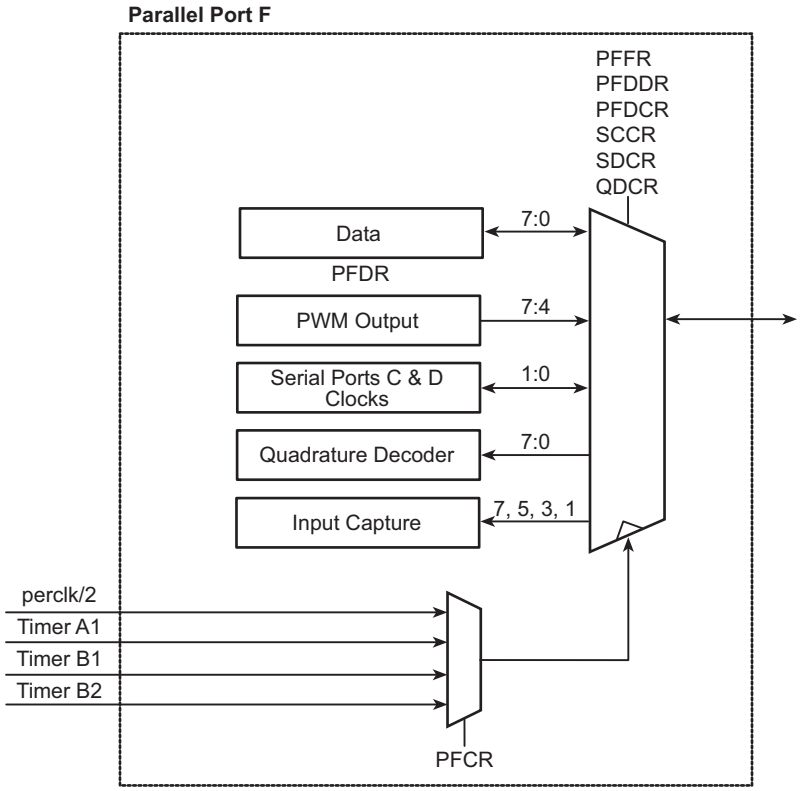
When used as outputs, the Parallel Port F bits are buffered, with the data written to PFDR transferred to the output pins on a selected timing edge. Either the peripheral clock or the outputs of Timer A1, Timer B1, or Timer B2 can be used for this function, with each nibble of the port having a separate select field to control this timing.. Each bit can either be programmed as an open-drain output or a standard output

These inputs and outputs are also used for access to other peripherals on the chip. As outputs, the Parallel Port F outputs can carry the four Pulse-Width Modulator (PWM) outputs. As inputs, Parallel Port F inputs can carry the inputs to the Quadrature Decoders. When Serial Port C or Serial Port D is used in the clocked serial mode, two pins of Parallel Port F are used to carry the serial clock signals. When the internal clock is selected in these serial ports, the corresponding bit of Parallel Port F is set as an output.

Table 13-1. Parallel Port F Alternate Pin Functions

Pin Name	Inputs			Outputs	
	Serial Ports A–D	Quadrature Decoder	Input Capture	PWM	Serial Ports A–D
PF7	—	AQD2A	×	PWM3	—
PF6	—	AQD2B	—	PWM2	—
PF5	—	AQD1A	×	PWM1	—
PF4	—	AQD1B	—	PWM0	—
PF3	—	QD2A	×	—	—
PF2	—	QD2B	—	—	—
PF1	CLKC	QD1A	×	—	CLKC
PF0	CLKD	QD1B	—	—	CLKD

13.1.1 Block Diagram



13.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port F Data Register	PFDR	0x0038	R/W	xxxxxxxx
Port F Control Register	PFCR	0x003C	W	xx00xx00
Port F Function Register	PFFR	0x003D	W	xxxxxxxx
Port F Drive Control Register	PFDCR	0X003E	W	xxxxxxxx
Port F Data Direction Register	PFDDR	0x003F	W	00000000

13.2 Dependencies

13.2.1 I/O Pins

Parallel Port F uses the pins PF0 through PF7. These pins can be used individually as data inputs or outputs, or as clocks for Serial Ports C and D; four of the Parallel Port F lines can be used for input capture or as Quadrature Decoder inputs.

All pins are set as inputs on startup, and the alternate pin functions are disabled.

13.2.2 Clocks

All outputs on Parallel Port F are clocked by the peripheral clock divided by two unless changed in PFCR, where the option of updating the Parallel Port F pins can be synchronized to the output of Timer A1, Timer B1, or Timer B2.

13.2.3 Other Registers

Register	Function
SCCR, SDCR	Configure a Parallel Port F pin as a clocked serial pin.
QDCR	Enable Quadrature Decoder inputs on Parallel Port F pins.

13.2.4 Interrupts

There are no interrupts associated with Parallel Port F.

13.3 Operation

The following steps must be taken before using Parallel Port F.

1. Select the desired input/output direction for each pin via PFDDR.
2. Use PFCR to select the clock.
3. If an alternative function is desired for a pin, select it via PFFR, PFDCR, SCCR, SDCR, or QDCR.

Once the port is set up, data can be read or written by accessing PEDR. The value of an output pin read in from PEDR will reflect its current output value, but any value written to an input pin will not appear until that pin becomes an output.

On reset, PFDDR is zeroed, making all the pins inputs. In addition, bits 0, 1, 4, and 5 in PFCR are zeroed to ensure that data are clocked into the output registers when loaded. All other registers associated with Parallel Port F are not initialized on reset.

NOTE: There may be a conflict in using Parallel Port A and Parallel Port F on the original Rabbit 3000 chip. Either Parallel Port A can be used as inputs, in which case Parallel Port F has full function, or if Parallel Port A cannot be used as inputs, use any pins on Parallel Port F not used for PWM or serial clock outputs as inputs and take the precaution of setting up Parallel Port F before the conflicting functionality of Parallel Port A is enabled. Refer to Rabbit's Technical Note TN228, ***Rabbit 3000 Parallel Port F Bug***, for more information.

13.4 Register Descriptions

Parallel Port F Data Register (PFDR) (Address = 0x0038)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port F pins PF7–PF0 is reported.
	Write	The Parallel Port F preload register is written with this value for transfer to the Parallel Port F output register on the next rising edge of the peripheral clock.

Parallel Port F Control Register (PFCR) (Address = 0x003C)		
Bit(s)	**Value**	**Description**
7:6	00	These bits are ignored and should be written with zero.
5:4	00	The upper nibble peripheral clock is perclk/2.
	01	The upper nibble peripheral clock is the output of Timer A1.
	10	The upper nibble peripheral clock is the output of Timer B1.
	11	The upper nibble peripheral clock is the output of Timer B2.
3:2	00	These bits are ignored and should be written with zero.
1:0	00	The lower nibble peripheral clock is perclk/2.
	01	The lower nibble peripheral clock is the output of Timer A1.
	10	The lower nibble peripheral clock is the output of Timer B1.
	11	The lower nibble peripheral clock is the output of Timer B2.

Parallel Port F Function Register (PFFR) (Address = 0x003D)		
Bit(s)	**Value**	**Description**
7:4	0	The corresponding port bit is a parallel port I/O pin.
	1	The corresponding port bit is a PWM output (PWM3–PWM0).
3:2	0	The corresponding port bit is a parallel port I/O pin.
	1	The corresponding port bit is a Quadrature Decoder input (QD2A–QD2B). (This value was not used for the Rabbit 3000 chip.).
1:0	0	The corresponding port bit is a parallel port I/O pin.
	1	The corresponding pin pair enables the clock ouputs for Serial Ports C and D when the serial port is configured for internal clock generation or is a Quadrature Decoder input (QD1A–QD1B).

Parallel Port F Drive Control Register (PFDCR) (Address = 0x003E)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit, as an output, is driven high and low.
	1	The corresponding port bit, as an output, is open-drain.

Parallel Port F Data Direction Register (PFDDR) (Address = 0x003F)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit is input.
	1	The corresponding port bit is an output.

Serial Port x Control Register (SCCR) (Address = 0x00E4) (SDCR) (Address = 0x00F4)		
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynch mode.
	01	In clocked serial mode, start a byte receive operation.
	10	In clocked serial mode, start a byte transmit operation.
	11	In clocked serial mode, start a byte transmit operation and a byte receive operation simultaneously.
5	0	Enable the receiver input.
	1	Disable the receiver input.
4	x	This bit is ignored.
3:2	00	Asynch mode with 8 bits per character.
	01	Asynch mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	Clocked serial mode with external clock. Serial Port C clock is on Parallel Port F pin 1 Serial Port D clock is on Parallel Port F pin 0
	11	Clocked serial mode with internal clock. Serial Port C clock is on Parallel Port F pin 1 Serial Port D clock is on Parallel Port F pin 0
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.
	11	The serial port uses Interrupt Priority 3.

Quad Decode Control Register (QDCR) (Address = 0x0091)		
Bit(s)	**Value**	**Description**
7:6	00	Disable Quadrature Decoder 2 inputs. Writing a new value to these bits will not cause Quadrature Decoder 2 to increment or decrement.
	01	This bit combination is reserved and should not be used.
	10	Quadrature Decoder 2 inputs from Port F bits 3 and 2.
	11	Quadrature Decoder 2 inputs from Port F bits 7 and 6.
5	0	Eight-bit Quadrature Decoder counters (both channels).
	1*	Ten-bit Quadrature Decoder counters (both channels).
4		This bit is reserved and should be written as zero.
3:2	00	Disable Quadrature Decoder 1 inputs. Writing a new value to these bits will not cause Quadrature Decoder 1 to increment or decrement.
	01	This bit combination is reserved and should not be used.
	10	Quadrature Decoder 1 inputs from Port F bits 1 and 0.
	11	Quadrature Decoder 1 inputs from Port F bits 5 and 4.
1:0	00	Quadrature Decoder interrupts are disabled.
	01	Quadrature Decoder interrupt use Interrupt Priority 1.
	10	Quadrature Decoder interrupt use Interrupt Priority 2.
	11	Quadrature Decoder interrupt use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip

14. PARALLEL PORT G

14.1 Overview

Parallel Port G is a byte-wide port with each bit programmable for data direction. These are simple inputs and outputs controlled and reported in the Port G Data Register (PGDR).

When used as outputs, the Parallel Port G bits are buffered, with the data written to PGDR transferred to the output pins on a selected timing edge. Either the peripheral clock or the outputs of Timer A1, Timer B1, or Timer B2 can be used for this function, with each nibble of the port having a separate select field to control this timing. Each bit can either be programmed as open-drain or driven high and low.

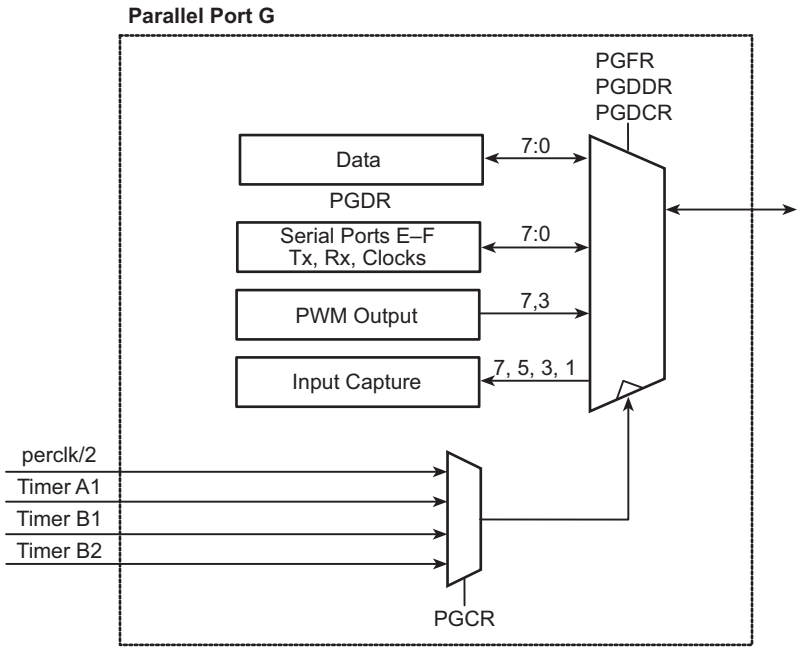
These inputs and outputs are also used for PWM outputs and for access to the data and clock I/O of SDLC/HDLC Serial Ports E and F.

Table 14-1. Parallel Port G Alternate Pin Functions

Pin Name	Inputs		Outputs	
	Serial Ports E–F	Input Capture	PWM	Serial Ports E–F
PG7	RXE	×	APWM1*	
PG6	—	—		TXE
PG5	RCLKE ARXE*	×		RCLKE
PG4	TCLKE ARCLKE*	—		TCLKE RTCLKE*
PG3	RXF	×	APWM0*	
PG2	—	—		TXF
PG1	RCLKF ARXF*	×		RCLKF
PG0	TCLKF ARCLKF*	—		TCLKF RTCLKF*

* Introduced with Rabbit 3000A chip.

14.1.1 Block Diagram



14.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Port G Data Register	PGDR	0x0048	R/W	xxxxxxxx
Port G Control Register	PGCR	0x004C	W	xx00xx00
Port G Function Register	PGFR	0x004D	W	xxxxxxxx
Port G Drive Control Register	PGDCR	0x004E	W	xxxxxxxx
Port G Data Direction Register	PGDDR	0x004F	W	00000000

14.2 Dependencies

14.2.1 I/O Pins

Parallel Port G uses the pins PG0 through PG7. These pins can be used individually as data inputs or outputs, or as clocked SDLC/HDLC Serial Ports E and F.

All pins are set as inputs on startup, and the alternate pin functions are disabled.

14.2.2 Clocks

All outputs on Parallel Port G are clocked by the peripheral clock divided by two unless changed in PGCR, where the option of updating the Parallel Port G pins can be synchronized to the output of Timer A1, Timer B1, or Timer B2.

14.2.3 Interrupts

There are no interrupts associated with Parallel Port G.

14.3 Operation

The following steps must be taken before using Parallel Port G.

1. Select the desired input/output direction for each pin via PGDDR.
2. Use PGCR to select the clock.
3. If an alternative function is desired for a pin, select it via PGFR.

Once the port is set up, data can be read or written by accessing PGDR. The value of an output pin read in from PGDR will reflect its current output value, but any value written to an input pin will not appear until that pin becomes an output.

On reset, PGDDR is zeroed, making all pins inputs, and disabling the alternate output functions. In addition, bits 0, 1, 4, and 5 in PGCR are zeroed to ensure that data are clocked into the output registers when loaded. All other registers associated with Parallel Port G are not initialized on reset.

14.4 Register Descriptions

Parallel Port G Data Register (PGDR) (Address = 0x0048)		
Bit(s)	**Value**	**Description**
7:0	Read	The current state of Parallel Port G pins PG7–PG0 is reported.
	Write	The Parallel Port G buffer is written with this value for transfer to the Parallel Port G output register on the next rising edge of the peripheral clock.

Parallel Port G Control Register (PGCR) (Address = 0x004C)		
Bit(s)	**Value**	**Description**
7:6	00	These bits are ignored and should be written with zero.
5:4	00	The upper nibble peripheral clock is perclk/2.
	01	The upper nibble peripheral clock is the output of Timer A1.
	10	The upper nibble peripheral clock is the output of Timer B1.
	11	The upper nibble peripheral clock is the output of Timer B2.
3:2	00	These bits are ignored and should be written with zero.
1:0	00	The lower nibble peripheral clock is perclk/2.
	01	The lower nibble peripheral clock is the output of Timer A1.
	10	The lower nibble peripheral clock is the output of Timer B1.
	11	The lower nibble peripheral clock is the output of Timer B2.

Parallel Port G Function Register (PGFR) (Address = 0x004D)		
Bit(s)	**Value**	**Description**
7	0	Parallel Port G pin 7 is always a parallel port input.
	1	Parallel Port G pin 7 is enabled as RxE.
6	0	Parallel Port G pin 6 is a parallel port output.
	1	Parallel Port G pin 6 drives TxE.
5	0	Parallel Port G pin 5 is always a parallel port input.
	1	Parallel Port G pin 5 is enabled as an SDLC/HDLC.receive clock output for Serial Port E.
4	0	Parallel Port G pin 4 is a parallel port output.
	1	Parallel Port G pin 4 is enabled as an SDLC/HDLC.transmit clock output for Serial Port E.
3	0	Parallel Port G pin 3 is always a parallel port input.
	1	Parallel Port G pin 3 is enabled as RxF.
2	0	Parallel Port G pin 2 is a parallel port output.
	1	Parallel Port G pin 2 drives TxF.
1	0	Parallel Port G pin 1 is always a parallel port input.
	1	Parallel Port G pin 1 is enabled as an SDLC/HDLC.receive clock output for Serial Port F.
0	0	Parallel Port G pin 0 is a parallel port output.
	1	Parallel Port G pin 0 is enabled as an SDLC/HDLC.transmit clock output for Serial Port F.

Parallel Port G Drive Control Register (PGDCR) (Address = 0x004E)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit, as an output, is driven high and low.
	1	The corresponding port bit, as an output, is open-drain.

Parallel Port G Data Direction Register (PGDDR) (Address = 0x004F)		
Bit(s)	**Value**	**Description**
7:0	0	The corresponding port bit is input.
	1	The corresponding port bit is an output.

15. TIMER A

15.1 Overview

The Timer A peripheral consists of ten separate eight-bit countdown timers, A1–A10. Each counter counts down from a programmed time constant, which is automatically reloaded into the respective counter when the count reaches zero. For example, if the reload register contains 127, then 128 pulses enter on the left before a pulse exits on the right (see Figure 15-1). If the reload register contains zero, then each pulse on the left results in a pulse on the right, that is, there is division by one. The reload register can contain any number in the range from 0 to 255. The counter divides by (n + 1).

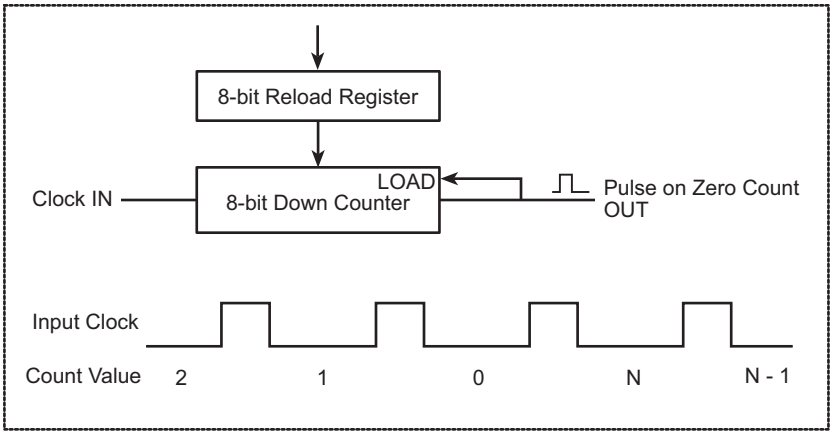


Figure 15-1. Reload Register Operation

For Timers A1–A7 the terminal count condition is reported in a status register and can be programmed to generate an interrupt. Six of these seven timers (A2–A7) have the option of being cascaded from Timer A1, but the primary clock for all of the timers is the peripheral clock either directly or divided by 2 (the default). The output pulses are always one clock wide. Clocking of the timers takes place on the negative edge of this pulse. When the counter reaches zero, the reload register is loaded into the counter on the next input pulse instead of a count being performed.

Timers A2–A7 can be used to generate baud rates for Serial Ports A–F, or they can be used as general-purpose timers if the dedicated timers on the Rabbit 3000 serial ports are used. The three remaining timers (A8–A10) serve as prescalers for the Input Capture, PWM, and Quadrature Decoder peripherals respectively. The peripherals clocked by these timers

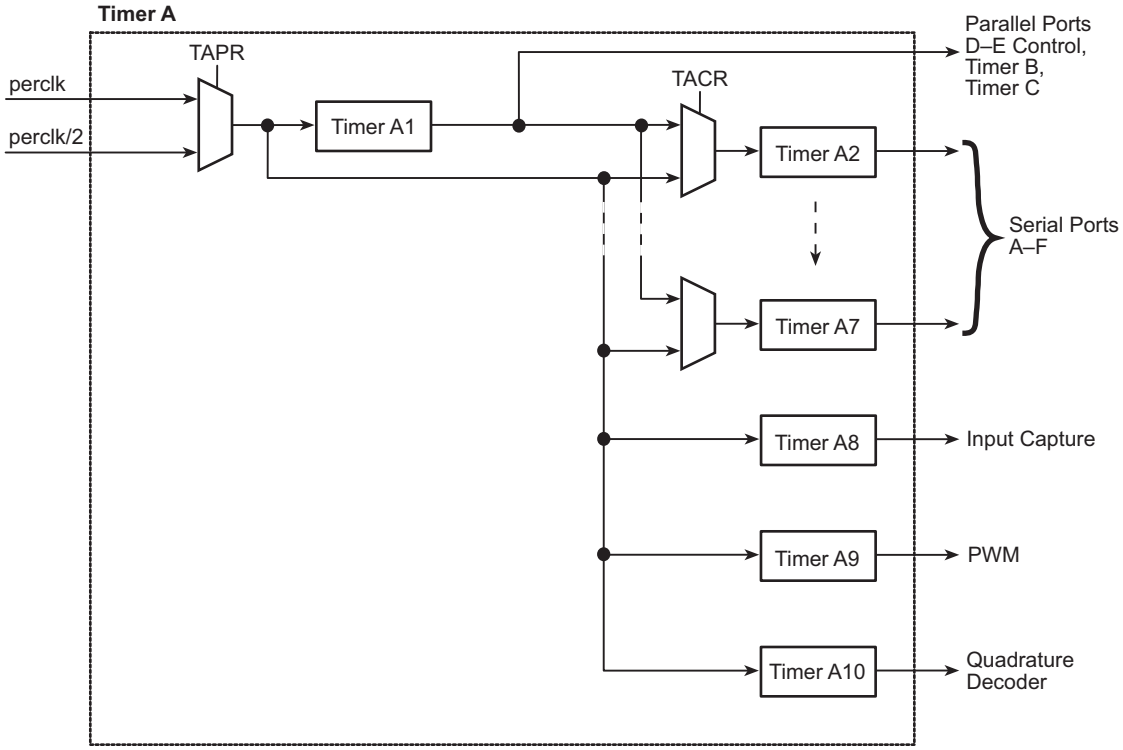
can generate interrupts, but the timers themselves cannot. Furthermore, these timers cannot be cascaded with Timer A1.

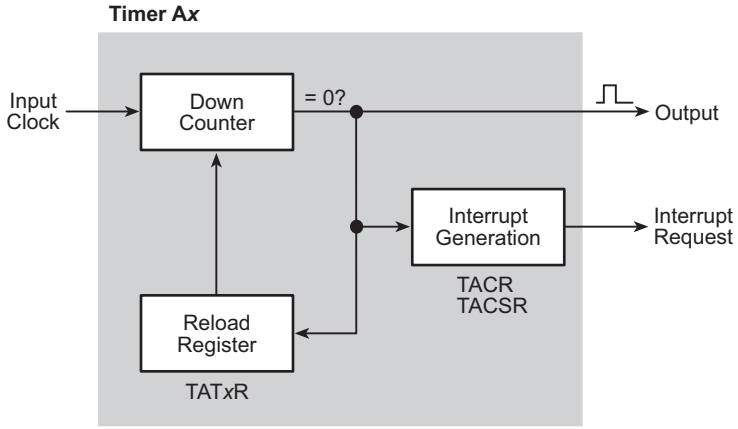
The individual Timer A capabilities are summarized in the table below. There is a bit in the control/status register to disable all ten timers globally.

Timer	Cascade from A1	Interrupt	Associated Peripheral
A1	No	Yes	Parallel Ports D–G, Timer B
A2	Yes	Yes	Serial Port E
A3	Yes	Yes	Serial Port F
A4	Yes	Yes	Serial Port A
A5	Yes	Yes	Serial Port B
A6	Yes	Yes	Serial Port C
A7	Yes	Yes	Serial Port D
A8	No	No	Input Capture
A9	No	No	Pulse-Width Modulator
A10	No	No	Quadrature Decoder

There is one interrupt vector for Timer A and a common interrupt priority. A common status register (TACSR) has bits for Timers A1–A7 that indicate if the output pulse for that timer has taken place since the last read of the status register. These bits are cleared when the status register is read. No bit will be lost. Either it will be read by the status register read or it will be set after the status register read is complete. If a bit is on and the corresponding interrupt is enabled, an interrupt will occur when priorities allow. However, a separate interrupt is not guaranteed for each bit with an enabled interrupt. If the bit is read in the status register, it is cleared and no further interrupt corresponding to that bit will be requested. It is possible that one bit will cause an interrupt, and then one or more additional bits will be set before the status register is read. After these bits are cleared, they cannot cause an interrupt. The proper rule to follow is for the interrupt routine to handle all bits that it sees set.

15.1.1 Block Diagram





15.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Timer A Control/Status Register	TACSR	0x00A0	R/W	00000000
Timer A Prescale Register	TAPR	0x00A1	W	xxxxxxx1
Timer A Time Constant 1 Register	TAT1R	0x00A3	W	xxxxxxxx
Timer A Control Register	TACR	0x00A4	W	00000000
Timer A Time Constant 2 Register	TAT2R	0x00A5	W	xxxxxxxx
Timer A Time Constant 8 Register	TAT8R	0x00A6	W	xxxxxxxx
Timer A Time Constant 3 Register	TAT3R	0x00A7	W	xxxxxxxx
Timer A Time Constant 9 Register	TAT9R	0x00A8	W	xxxxxxxx
Timer A Time Constant 4 Register	TAT4R	0x00A9	W	xxxxxxxx
Timer A Time Constant 10 Register	TAT10R	0x00AA	W	xxxxxxxx
Timer A Time Constant 5 Register	TAT5R	0x00AB	W	xxxxxxxx
Timer A Time Constant 6 Register	TAT6R	0x00AD	W	xxxxxxxx
Timer A Time Constant 7 Register	TAT7R	0x00AF	W	xxxxxxxx

15.2 Dependencies

15.2.1 I/O Pins

The output of Timer A does not come out directly on any of the I/O pins. It can be used to control when the output occurs on Parallel Ports D–G, and can affect the output times of Serial Ports A–F and the PWM. Timer A also affects the resolution of the Quadrature Decoder and the input capture inputs.

15.2.2 Clocks

The timers in Timer A can be clocked by either perclk or perclk/2, as selected in TAPR. In addition, Timers A2–A7 can be clocked by the output of Timer A1 by selecting that option in TACSR.

15.2.3 Other Registers

Register	Function
GCSR	Select peripheral clock mode.

15.2.4 Interrupts

A Timer A interrupt can be generated whenever Timers A1–A7 decrement to zero by enabling the appropriate bit in TACSR. The interrupt request is cleared when TACSR is read.

The Timer A interrupt vector is in the IIR at offset 0x0A0. It can be set as Priority 1, 2, or 3 in TACR.

15.3 Operation

The following steps explain how to set up a Timer A timer.

1. Select perclk as the Timer A input clock in TAPR (default is perclk/2).
2. Select the source clocks for Timers A2–A7 in TACR.
3. Write the desired divider value to TATxR for all timers that will be used.
4. Enable Timer A by writing a 1 to bit 0 of TACSR.

15.3.1 Handling Interrupts

The following steps explain how an interrupt is set up and used. Remember to set up the interrrupt vector *before* you enable the interrupts.

1. Write the vector of the interrupt service routine to the internal interrupt table.
2. Configure TACSR to select which timers will generate an interrupt.
3. Configure TACR to select the interrupt priority (note that interrupts will be enabled once this value is set). This should be done last.

The interrupt request is cleared by reading from TACSR.

15.3.2 Example ISR

A sample interrupt handler is shown below.

```
timerA_isr::
   push af              ; save used registers
   ioi ld a, (TACSR)    ; clear the interrupt request and get status

   ; handle all interrupts flagged in TACSR here

   pop af               ; restore registers
   ipres
   ret
```

15.4 Register Descriptions

Timer A Control/Status Register (TACSR) (Address = 0x00A0)		
Bit(s)	**Value**	**Description**
7:1 (Read-only)	0	The corresponding Timer A counter has not reached its terminal count.
	1	The corresponding Timer A counter has reached its terminal count. These status bits (not the interrupt enable bits) are cleared by the read of this register, as is the Timer A interrupt.
7:1 (Write-only)	0	The corresponding Timer A interrupt is disabled.
	1	The corresponding Timer A interrupt is enabled.
0 (Write-only)	0	The clock input for Timer A is disabled.
	1	The clock input for Timer A is enabled.

Timer A Prescale Register (TAPR) (Address = 0x00A1)		
Bit(s)	**Value**	**Description**
7:1		These bits are reserved and should be written with zero.
0	0	The main clock for Timer A is the peripheral clock (perclk).
	1	The main clock for Timer A is the peripheral clock divided by two (perclk/2).

Timer A Control Register	(TACR)	(Address = 0x00A4)
Bit(s)	**Value**	**Description**
7	0	Timer A7 clocked by the main Timer A clock.
	1	Timer A7 clocked by the output of Timer A1.
6	0	Timer A6 clocked by the main Timer A clock.
	1	Timer A6 clocked by the output of Timer A1.
5	0	Timer A5 clocked by the main Timer A clock.
	1	Timer A5 clocked by the output of Timer A1.
4	0	Timer A4 clocked by the main Timer A clock.
	1	Timer A4 clocked by the output of Timer A1.
3	0	Timer A3 clocked by the main Timer A clock.
	1	Timer A3 clocked by the output of Timer A1.
2	0	Timer A2 clocked by the main Timer A clock.
	1	Timer A2 clocked by the output of Timer A1.
1:0	00	Timer A interrupts are disabled.
	01	Timer A interrupt use Interrupt Priority 1.
	10	Timer A interrupt use Interrupt Priority 2.
	11	Timer A interrupt use Interrupt Priority 3.

Timer A Time Constant x Register	(TAT1R) (TAT2R) (TAT3R) (TAT4R) (TAT5R) (TAT6R) (TAT7R) (TAT8R) (TAT9R) (TAT10R)	(Address = 0x00A3) (Address = 0x00A5) (Address = 0x00A7)) (Address = 0x00A9) (Address = 0x00AB) (Address = 0x00AD) (Address = 0x00AF) (Address = 0x00A6) (Address = 0x00A8) (Address = 0x00AA)
Bit(s)	**Value**	**Description**
7:0		Time constant for the Timer A counter. This time constant will take effect the next time that the Timer A counter counts down to zero. The timer counts modulo $n + 1$, where n is the programmed time constant.

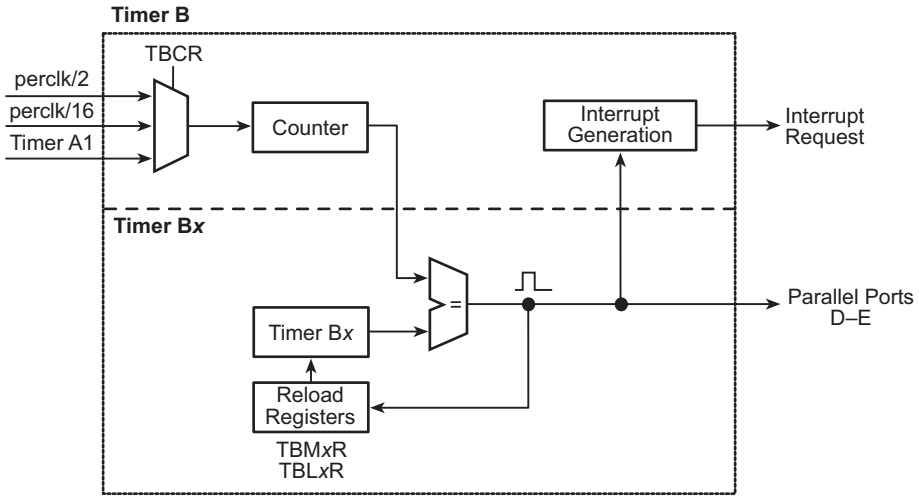
Global Control/Status Register (GCSR) (Address = 0x0000)		
Bit(s)	**Value**	**Description**
4:2	000	Processor clock from the main clock, divided by eight. Peripheral clock from the main clock, divided by eight.
	001	Processor clock from the main clock, divided by eight. Peripheral clock from the main clock.
	010	Processor clock from the main clock. Peripheral clock from the main clock.
	011	Processor clock from the main clock, divided by two. Peripheral clock from the main clock, divided by two.
	100	Processor clock from the 32 kHz clock, optionally divided via GPSCR. Peripheral clock from the 32 kHz clock, optionally divided via GPSCR.
	101	Processor clock from the 32 kHz clock, optionally divided via GPSCR. Peripheral clock from the 32 kHz clock, optionally divided via GPSCR. The fast clock is disabled.
	110	Processor clock from the main clock, divided by four. Peripheral clock from the main clock, divided by four.
	111	Processor clock from the main clock, divided by six. Peripheral clock from the main clock, divided by six.

16. TIMER B

16.1 Overview

The Timer B peripheral consists of a ten-bit free running up-counter and two match registers. Timer B is driven by perclk/2, by perclk/16, or by the output of Timer A1. Timer B generates an output pulse whenever the counter reaches the match value. This output pulse can generate an interrupt and will set a status bit in the status register. The processor may then write a new value to the match register. This allows Timer B to be used for pulse-width or pulse-position modulation because the outputs of Timer B can clock the outputs on Parallel Ports D and E.

16.1.1 Block Diagram



16.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Timer B Control/Status Register	TBCSR	0x00B0	R/W	xxxxx000
Timer B Control Register	TBCR	0x00B1	W	xxxx0000
Timer B MSB 1 Register	TBM1R	0x00B2	W	xxxxxxxx
Timer B LSB 1 Register	TBL1R	0x00B3	W	xxxxxxxx
Timer B MSB 2 Register	TBM2R	0x00B4	W	xxxxxxxx
Timer B LSB 2 Register	TBL2R	0x00B5	W	xxxxxxxx
Timer B Count MSB Register	TBCMR	0x00BE	R	xxxxxxxx
Timer B Count LSB Register	TBCLR	0x00BF	R	xxxxxxxx

16.2 Dependencies

16.2.1 I/O Pins

The output of Timer B does not come out directly on any of the I/O pins. It can be used to control when the output occurs on Parallel Ports D–E.

16.2.2 Clocks

The timer in Timer B can be clocked by perclk/2, perclk/16, or by countdown Timer A1 as selected in TBCR.

16.2.3 Other Registers

Register	Function
GCSR	Select peripheral clock mode.

16.2.4 Interrupts

A Timer B interrupt can be generated whenever the counter equals one of the match registers by enabling the appropriate bit in TBCSR. The interrupt request is cleared when TBCSR is read.

16.3 Operation

The following steps explain how to set up a Timer B countdown timer.

1. Select perclk/2, perclk/16, or countdown Timer A1 in TBCR.
2. Enable Timer B by writing a 1 to bit 0 of TBCSR.

16.3.1 Handling Interrupts

The following steps explain how an interrupt is set up and used.

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Configure TBCSR to select which match registers will generate an interrupt.
3. Configure TBCR to select the interrupt priority (note that interrupts will be enabled once this value is set; this step should be done last).

The interrupt request is cleared by reading from TBCSR.

16.3.2 Example ISR

A sample interrupt handler is shown below.

```
timerB_isr::
   push af              ; save used registers
   ioi ld a, (TBCSR)    ; clear the interrupt request and get status

   ; handle all interrupts flagged in TBCSR here
   ; reload match register(s) if necessary

   pop af               ; restore used registers
   ipres
   ret
```

16.4 Register Descriptions

Timer B Control/Status Register (TBCSR) (Address = 0x00B0)		
Bit(s)	**Value**	**Description**
7:3		These bits always read as zero.
2:1 (Read-only)	0	The corresponding Timer B comparator has not encountered a match condition.
	1	The corresponding Timer B comparator has encountered a match condition. These status bits (but not the interrupt enable bits) are cleared by the read of this register, as is the Timer B interrupt.
2:1 (Write-only)	0	The corresponding Timer B interrupt is disabled.
	1	The corresponding Timer B interrupt is enabled.
0 (Write-only)	0	The clock input for Timer B is disabled.
	1	The clock input for Timer B is enabled.

Timer B Control Register (TBCR) (Address = 0x00B1)		
Bit(s)	**Value**	**Description**
7:4		These bits are reserved and should be written with zero.
3:2	00	Timer B clocked by main Timer B clock (perclk/2).
	01	Timer B clocked by the output of Timer A1.
	10	Timer B clocked by the perpheral clock divided by 16.
	11	Timer B clocked by the perpheral clock divided by 16.
1:0	00	Timer B interrupts are disabled.
	01	Timer B interrupt use Interrupt Priority 1.
	10	Timer B interrupt use Interrupt Priority 2.
	11	Timer B interrupt use Interrupt Priority 3.

Timer B Count MSB x Register	(TBM1R) (TBM2R)	(Address = 0x00B2) (Address = 0x00B4)
Bit(s)	**Value**	**Description**
7:6	Write	Two MSBs of the compare value for the Timer B comparator. This compare value will be loaded into the actual comparator when the current compare detects a match.
5:0		These bits are always read as zeroes.

Timer B Count LSB x Register	(TBL1R) (TBL2R)	(Address = 0x00B3) (Address = 0x00B5)
Bit(s)	**Value**	**Description**
7:0	Write	Eight LSBs of the compare value for the Timer B comparator are stored. This compare value will be loaded into the actual comparator when the current compare detects a match.

Timer B Count MSB Register	(TBCMR)	(Address = 0x00BE)
Bit(s)	**Value**	**Description**
7:6	Read	The current value of the two MSBs of the Timer B counter are reported.
5:0		These bits are always read as zeros.

Timer B Count LSB Register	(TBCLR)	(Address = 0x00BF)
Bit(s)	**Value**	**Description**
7:0	Read	The current value of the eight LSBs of the Timer B counter are reported.

Global Control/Status Register (GCSR) (Address = 0x0000)		
Bit(s)	**Value**	**Description**
4:2	000	Processor clock from the main clock, divided by eight. Peripheral clock from the main clock, divided by eight.
	001	Processor clock from the main clock, divided by eight. Peripheral clock from the main clock.
	010	Processor clock from the main clock. Peripheral clock from the main clock.
	011	Processor clock from the main clock, divided by two. Peripheral clock from the main clock, divided by two.
	100	Processor clock from the 32 kHz clock, optionally divided via GPSCR. Peripheral clock from the 32 kHz clock, optionally divided via GPSCR.
	101	Processor clock from the 32 kHz clock, optionally divided via GPSCR. Peripheral clock from the 32 kHz clock, optionally divided via GPSCR. The fast clock is disabled.
	110	Processor clock from the main clock, divided by four. Peripheral clock from the main clock, divided by four.
	111	Processor clock from the main clock, divided by six. Peripheral clock from the main clock, divided by six.

17. SERIAL PORTS A – D

17.1 Overview

Serial Ports A, B, C, and D are identical, except for the source of the data clock and the transmit, receive, and clock pins. Serial Port A is special because it can be used to bootstrap the processor. Each serial port can be used in the asynchronous or the clocked serial mode with an internal or external clock.

In the asynchronous mode, either 7 or 8 data bits can be transferred, and a parity bit and/or an additional address (0) or long stop (1) bit can be appended as well. Parity and the address/long stop bits are also detected when they are received. The asynchronous mode is full-duplex, while the clocked mode can be half or full-duplex.

Both transmit and receive have one byte of buffering — a byte may be read while another byte is being received, or the next byte to be transmitted can be loaded while the current byte is still being transferred out. The byte is available in the buffer after the final bit is sampled.

The status of each serial port is available in the Serial Port Status Registers (SxSR), and contains information on whether a received byte is available, the receive buffer was overrun, a parity error was received, and the transmit buffer is empty or busy sending a byte. The status is updated when the final bit of a received byte is sampled, or when the final bit of a transmitted byte is sent out. Each serial port has a separate interrupt vector that will be requested whenever the transmit buffer is emptied or the receive buffer contains a full byte.

All four common SPI clock modes are supported, and the bit order of the data may be either MSB or LSB first. The transmit and receive operations are under program control as well.

> **NOTE:** The maximum external serial clock rate for the SPI port is 1/6th of the peripheral clock rate.

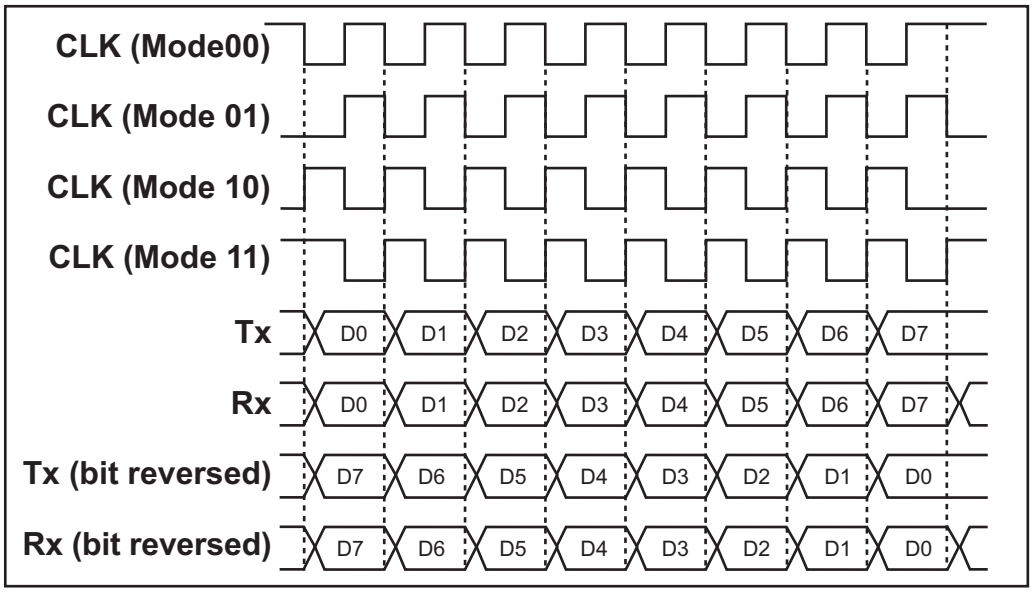


Figure 17-1. Serial Ports A – D Operation in Clocked Serial Mode

In the asynchronous mode, IrDA-compliant RZI encoding can be enabled to reduce the bit widths to 3/16 the normal width (1/8 the normal width if the serial data clock is 8× instead of 16×), which allows the serial port signal to be connected directly to an IrDA transceiver.

It is possible to synchronize a clocked serial transfer to the match registers of Timer B to generate precisely timed transmissions.

Table 17-1. Timer A Data Clocks

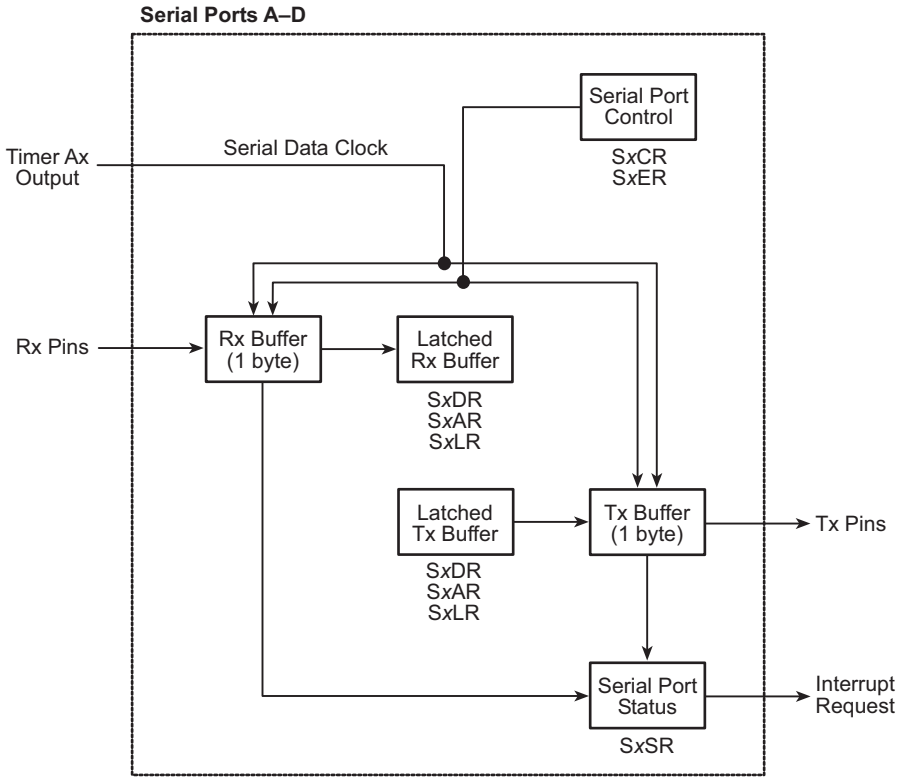
Serial Port	Data Clock
A	Timer A4
B	Timer A5
C	Timer A6
D	Timer A7

The serial port data clocks can be generated from the appropriate 8-bit timer from Timer A shown in Table 17-1. The resulting bit data rate in the asynchronous mode is 1/8 or 1/16 the data clock rate (selectable). However,the bit data rate in the clocked serial mode is equal to the data clock rate as generated from the appropriate Timer A timer.

When Serial Port A is used in the asynchronous bootstrap mode, the 32 kHz clock is used to generate the expected 2400 bps data rate. An external clock must be supplied for the clocked serial bootstrap mode.

The behavior of the serial port during a break (line held low) is configurable; character assembly can continue during the break condition to allow for timing the break, or character assembly can be inhibited to reduce the interrupt overhead.

17.1.1 Block Diagram



17.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port A Data Register	SADR	0x00C0	R/W	xxxxxxxx
Serial Port A Address Register	SAAR	0x00C1	W	xxxxxxxx
Serial Port A Long Stop Register	SALR	0x00C2	W	xxxxxxxx
Serial Port A Status Register	SASR	0x00C3	R	0xx00000
Serial Port A Control Register	SACR	0x00C4	W	xx000000
Serial Port A Extended Register	SAER	0x00C5	R/W	00000000
Serial Port B Data Register	SBDR	0x00D0	R/W	xxxxxxxx
Serial Port B Address Register	SBAR	0x00D1	W	xxxxxxxx
Serial Port B Long Stop Register	SBLR	0x00D2	W	xxxxxxxx
Serial Port B Status Register	SBSR	0x00D3	R	0xx00000
Serial Port B Control Register	SBCR	0x00D4	W	xx000000
Serial Port B Extended Register	SBER	0x00D5	R/W	00000000
Serial Port C Data Register	SCDR	0x00E0	R/W	xxxxxxxx
Serial Port C Address Register	SCAR	0x00E1	W	xxxxxxxx
Serial Port C Long Stop Register	SCLR	0x00E2	W	xxxxxxxx
Serial Port C Status Register	SCSR	0x00E3	R	0xx00000
Serial Port C Control Register	SCCR	0x00E4	W	xx000000
Serial Port C Extended Register	SCER	0x00E5	R/W	00000000
Serial Port D Data Register	SDDR	0x00F0	R/W	xxxxxxxx
Serial Port D Address Register	SDAR	0x00F1	W	xxxxxxxx
Serial Port D Long Stop Register	SDLR	0x00F2	W	xxxxxxxx
Serial Port D Status Register	SDSR	0x00F3	R	0xx00000
Serial Port D Control Register	SDCR	0x00F4	W	xx000000
Serial Port D Extended Register	SDER	0x00F5	R/W	00000000

17.2 Dependencies

17.2.1 I/O Pins

Serial Port A can transmit on parallel port pins PC6 or PD6, and can receive on pins PC7 or PD7. If the clocked serial mode is enabled, the serial clock is either transmitted or received on PB1. When an internal clock is selected in the clocked serial mode, PB1 is automatically enabled as a clock output.

Serial Port B can transmit on parallel port pins PC4 or PD4, and can receive on pins PC5 or PD5. If the clocked serial mode is enabled, the serial clock is either transmitted or received on PB0. When an internal clock is selected in the clocked serial mode, PB0 is automatically enabled as a clock output.

Serial Port C can transmit on parallel port pin PC2, and can receive on pin PC3. If the clocked serial mode is enabled, the serial clock will be on PF1.

Serial Port D can transmit on parallel port pin PC0, and can receive on pin PC1. If the clocked serial mode is enabled, the serial clock will be transmitted on PF0.

Table 17-2. Pin Usage Serial Ports A – D

Function	Serial Port A	Serial Port B	Serial Port C	Serial Port D
Transmit	PC6, PD6	PC4, PD4	PC2	PC0
Receive	PC7, PD7	PC5, PD5	PC3	PC1
Transmit Clock	PB1	PB0	PF1	PF0
Receive Clock	PB1	PB0	PF1	PF0

17.2.2 Clocks

The data clocks for Serial Ports A – D are based on the corresponding Timer A output and can be divided by a Timer A divider. The overall clock divider will be the value in the appropriate register plus one.

17.2.3 Other Registers

Register	Function
TAT4R	Time constant for Serial Port A
TAT5R	Time constant for Serial Port B
TAT6R	Time constant for Serial Port C
TAT7R	Time constant for Serial Port D
PCFR, PDFR	Alternate port output selection

17.2.4 Interrupts

A serial port interrupt can be generated whenever a byte is available in the receive buffer or when a byte is finished being transmitted out of the transmit buffer.

The serial port interrupt vectors are located in the IIR as follows.

- Serial Port A at offset 0x0C0
- Serial Port B at offset 0x0D0
- Serial Port C at offset 0x0E0
- Serial Port D at offset 0x0F0

Each of them can be set as Priority 1, 2, or 3 in SxCR, where x is A – D for the four serial ports.

17.3 Operation

17.3.1 Asynchronous Mode

The following steps explain how to set up Serial Ports A – D for asynchronous operation. The serial ports can be used by polling the status byte, but their performance will be better with an interrupt. These instructions also apply to the asynchronous operation of Serial Ports E – F.

1. Write the interrupt vector for the interrupt service routine to the internal interrupt table.
2. Set up the desired transmit pin by writing to the appropriate parallel port function register (PCFR or PDFR).
3. Select the appropriate mode by writing to SxCR (receive input port and 7 or 8 bits). Also select the interrupt priority.
4. Select additional options by writing to SxER (parity, RZI encoding, clock polarity, and behavior during break).
5. Write the desired divider value to TATxR for the appropriate serial port.

A sample asynchronous serial interrupt handler is shown below for Serial Port A.

```
async_sera_isr::
   push af              ; save used registers
   ioi ld a, (SASR)     ; get status
   bit 7,a              ; check if byte ready in RX buffer
   push af              ; save status for next check
   jr  z, check_for_tx
rx_ready:
   ioi ld a, (SADR)     ; read byte and clear interrupt

   ; do something with byte here

check_for_tx:
   pop af
   bit 3,a              ; check if TX buffer was emptied
   jr nz, done

   ; get next byte to be transmitted into A here

   ioi ld (SADR), a     ; load next byte into TX buffer and clear interrupt
done:
   pop af               ; restore used registers
   ipres
   ret
```

To transmit with an address (1) bit appended, write the data to SxAR instead of SxDR; to append a long stop (0) bit write to SxLR instead.

17.3.2 Clocked Serial Mode

The following steps explain how to set up Serial Ports A – D for the clocked serial mode. When the internal clock is selected, the Rabbit 3000 is in control of all transmit and receive operations. When an external clock is selected, the other device controls all transmit and receive operation. For both situations the decision between polling and interrupt-driven methods is application-dependent.

1. Write the interrupt vector for the interrupt service routine to the internal interrupt table.
2. Set up the desired data transmit and clock pins by writing to the appropriate parallel port function register (PCFR or PDFR).
3. Select the appropriate mode by writing to SxCR (receive input port and clock source). Also select the interrupt priority.
4. Select additional options by writing to SxER (clock polarity, bit order, and clock source if external).
5. Write the desired divider value to TATxR for the appropriate serial port.
6. There are two methods to transfer a byte:

   write the byte to SxDR and then write 10 (or 11) to bits 6–7 of SxCR to enable the transfer;
   write the byte to SxAR which will automatically start the transfer.

   If the internal clock is selected, the transmission will begin immediately; if an external clock is selected, the transmission will begin when the clock is detected.
7. To receive a byte, write 01 to bits 6-7 of SxCR to start the receive operation. If the internal clock is selected, the clock will begin immediately and the data will be read; if an external clock is selected, the receive will occur when the clock is detected.

A sample clocked serial interrupt handler is shown below for Serial Port B.

```
clocked_serb_isr::
   push af              ; save used registers
   ioi ld a, (SASR)     ; get status
   bit 7,a              ; check if byte ready in RX buffer
   push af              ; save status for next check
   jr  z, check_for_tx
rx_ready:
   ioi ld a, (SADR)     ; read byte and clear interrupt

   ; do something with received byte here

   ld a, 0x41           ; set bits 6-7 to 01, the other bits should
                        ; represent the desired SACR setup

   ioi ld (SACR), a     ; start a new receive operation

check_for_tx:
   pop af
   bit 3,a              ; check if TX buffer was emptied
   jr nz, done

   ; get next byte to be transmitted into A here

   ioi ld (SADR), a     ; load TX buffer with next byte and clear interrupt
done:
   pop af               ; restore used registers
   ipres
   ret
```

17.4 Register Descriptions

Serial Port x Data Register		(SADR) (Address = 0x00C0) (SBDR) (Address = 0x00D0) (SCDR) (Address = 0x00E0) (SDDR) (Address = 0x00F0)
Bit(s)	**Value**	**Description**
7:0	Read	Returns the contents of the receive buffer.
	Write	Loads the transmit buffer with a data byte for transmission.

Serial Port x Address Register		(SAAR) (Address = 0x00C1) (SBAR) (Address = 0x00D1) (SCAR) (Address = 0x00E1) (SDAR) (Address = 0x00F1)
Bit(s)	**Value**	**Description**
7:0	Read	Returns the contents of the receive buffer. Reading the data from this register in the clocked serial mode automatically causes the receiver to start a byte-receive operation, eliminating the need for software to issue the start-receive command.
	Write	Loads the transmit buffer with an address byte, marked with a “zero” address bit, for transmission. Writing the data to this register in the clocked serial mode causes the transmitter to start a byte-transmit operation, eliminating the need for the software to issue the start-transmit command.

Serial Port x Long Stop Register		(SALR) (Address = 0x00C2) (SBLR) (Address = 0x00D2) (SCLR) (Address = 0x00E2) (SDLR) (Address = 0x00F2)
Bit(s)	**Value**	**Description**
7:0	Read	Returns the contents of the receive buffer.
	Write	Loads the transmit buffer with an address byte, marked with a “one” address bit, for transmission.

Serial Port x Status Register (Asynchronous Mode Only)		(SASR) (Address = 0x00C3) (SBSR) (Address = 0x00D3) (SCSR) (Address = 0x00E3) (SDSR) (Address = 0x00F3)
Bit(s)	**Value**	**Description**
7	0	The receive data register is empty—no input character is ready.
	1	There is a byte in the receive buffer. The transition from “0” to “1” sets the receiver interrupt request flip-flop. The interrupt FF is cleared when the character is read from the data buffer. The interrupt FF will be immediately set again if there are more characters available in the FIFO or shift register to be transferred into the data buffer.
6	0	The byte in the receive buffer is data, received with a valid stop bit.
	1	The address bit or 9th (8th) bit received. This bit is set if the character in the receiver data register has a 9th (8th) bit. This bit is cleared and should be checked before reading a data register since a new data value with a new address bit may be loaded immediately when the data register is read. The byte in the receive buffer is an address, or a byte with a framing error. If an address bit is not expected. If the data in the buffer is all zeros, this may be a break.
5	0	The receive buffer was not overrun.
	1	The receive buffer was overrun. This bit is cleared by reading the receive buffer.
4	0	This bit is always zero in the asynchronous mode
3	0	The transmit buffer is empty.
	1	The transmit data buffer is full. This bit is set when the transmit data register is full, that is, a byte is written to the serial port data register. It is cleared when a byte is transferred to the transmitter shift register or FIFO, or a write operation is performed to the serial port status register. This bit will request an interrupt on the transition from 1 to 0 if interrupts are enabled. Transmit interrupts are cleared when the transmit buffer is written, or any value (which will be ignored) is written to this register.
2	0	The transmitter is idle.
	1	Transmit busy bit. This bit is set if the transmit shift register is busy sending data. It is set on the falling edge of the start bit, which is also the clock edge that transfers data from the transmit data register to the transmit shift register. The transmit busy bit is cleared at the end of the stop bit of the character sent. This bit will cause an interrupt to be latched when it goes from busy to not busy status after the last character has been sent (there are no more data in the transmit data register).
1:0	00	These bits are always zero in the asynchronous mode.

Serial Port x Status Register (Clocked Serial Mode Only)		(SASR) (Address = 0x00C3) (SBSR) (Address = 0x00D3) (SCSR) (Address = 0x00E3) (SDSR) (Address = 0x00F3)
Bit(s)	**Value**	**Description**
7	0	The receive data register is empty
	1	There is a byte in the receive buffer. The serial port will request an interrupt while this bit is set. The interrupt is cleared when the receive buffer is empty.
6	0	This bit is always zero in the clocked serial mode.
5	0	The receive buffer was not overrun.
	1	The receive buffer was overrun. This bit is cleared by reading the receive buffer.
4	0	This bit is always zero in the clocked serial mode.
3	0	The transmit buffer is empty.
	1	The transmit buffer is not empty. The serial port will request an interrupt when the transmitter takes a byte from the transmit buffer. Transmit interrupts are cleared when the transmit buffer is written, or any value (which will be ignored) is written to this register.
2	0	The transmitter is idle.
	1	The transmitter is sending a byte. An interrupt is generated when the transmitter clears this bit, which occurs only if the transmitter is ready to start sending another byte and the transmit buffer is empty.
1:0	00	These bits are always zero in the clocked serial mode.

Serial Port x Control Register		(SACR) (Address = 0x00C4) (SBCR) (Address = 0x00D4)
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynchronous mode.
	01	In clocked serial mode, start a byte receive operation.
	10	In clocked serial mode, start a byte transmit operation.
	11	In clocked serial mode, start a byte transmit operation and a byte receive operation simultaneously.
5:4	00	Parallel Port C is used for input.
	01	Parallel Port D is used for input.
	1x	Disable the receiver input.
3:2	00	Asynchronous mode with 8 bits per character.
	01	Asynchronous mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	Clocked serial mode with external clock. Serial Port A clock is on Parallel Port B pin 1 Serial Port B clock is on Parallel Port B pin 0
	11	Clocked serial mode with internal clock. Serial Port A clock is on Parallel Port B pin 1 Serial Port B clock is on Parallel Port B pin 0
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.

Serial Port x Control Register (SCCR) (Address = 0x00E4) (SDCR) (Address = 0x00F4)		
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynchronous mode.
	01	In clocked serial mode, start a byte receive operation.
	10	In clocked serial mode, start a byte transmit operation.
	11	In clocked serial mode, start a byte transmit operation and a byte receive operation simultaneously.
5	0	Enable the receiver input.
	1	Disable the receiver input.
4	x	This bit is ignored.
3:2	00	Asynchronous mode with 8 bits per character.
	01	Asynchronous mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	Clocked serial mode with external clock. Serial Port C clock is on Parallel Port F pin 1 Serial Port D clock is on Parallel Port F pin 0
	11	Clocked serial mode with internal clock. Serial Port C clock is on Parallel Port F pin 1 Serial Port D clock is on Parallel Port F pin 0
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.
	11	The serial port uses Interrupt Priority 3.

Serial Port x Extended Register (Asynchronous Mode Only)		(SAER) (Address = 0x00C5) (SBER) (Address = 0x00D5) (SCER) (Address = 0x00E5) (SDER) (Address = 0x00F5)
Bit(s)	**Value**	**Description**
7:5	xxx	These bits are ignored in the asynchronous mode and should be set to zeros.
4	0	Normal asynchronous data encoding.
	1	Enable RZI coding (3/16 bit cell IrDA-compliant).
3	0	Normal break operation. This option should be selected when address bits are expected.
	1	Fast break termination. At the end of break, a dummy character is written to the buffer, and the receiver can start character assembly after one bit time.
2	0	Asynchronous clock is 16× data rate.
	1	Asynchronous clock is 8× data rate.
1:0	xx	These bits are ignored in the asynchronous mode.

Serial Port x Extended Register (Clocked Serial Mode Only)		(SAER) (SBER) (SCER) (SDER)	(Address = 0x00C5) (Address = 0x00D5) (Address = 0x00E5) (Address = 0x00F5)
Bit(s)	**Value**	**Description**	
7	0	Normal clocked serial operation.	
	1	Timer-synchronized clocked serial operation.	
6	0	Timer-synchronized clocked serial uses Timer B1.	
	1	Timer-synchronized clocked serial uses Timer B2.	
5:4	00	Normal clocked serial clock polarity, inactive high. internal or external clock.	
	01	Normal clocked serial clock polarity, inactive low. internal clock only.	
	10	Inverted clocked serial clock polarity, inactive low. internal or external clock.	
	11	Inverted clocked serial clock polarity, inactive high. internal clock only.	
3:2	xx	These bits are ignored in the clocked serial mode and should be set to zeros.	
1	0	No effect on transmitter.	
	1	Terminate current clocked serial transmission. No effect on buffer.	
0	0	No effect on receiver.	
	1	Terminate current clocked serial reception.	

Timer A Time Constant x Register		(TAT4R) (TAT5R) (TAT6R) (TAT7R)	(Address = 0x00A9) (Address = 0x00AB) (Address = 0x00AD) (Address = 0x00AF)
Bit(s)	**Value**	**Description**	
7:0		Time constant for the Timer A counter. This time constant will take effect the next time that the Timer A counter counts down to zero. The timer counts modulo $n + 1$, where n is the programmed time constant.	

Parallel Port C Function Register (PCFR) (Address = 0x0055)		
Bit(s)	**Value**	**Description**
7	x	Parallel Port C pin 7 is always a parallel port input.
6	0	Parallel Port C pin 6 is a parallel port output.
	1	Parallel Port C pin 6 drives TxA.
5	x	Parallel Port C pin 5 is always a parallel port input.
4	0	Parallel Port C pin 4 is a parallel port output.
	1	Parallel Port C pin 4 drives TxB.
3	x	Parallel Port C pin 3 is always a parallel port input.
2	0	Parallel Port C pin 2 is a parallel port output.
	1	Parallel Port C pin 2 drives TxC.
1	x	Parallel Port C pin 1 is always a parallel port input.
0	0	Parallel Port C pin 0 is a parallel port output.
	1	Parallel Port C pin 0 drives TxD.

Parallel Port D Function Register (PDFR) (Address = 0x0065)		
Bit(s)	**Value**	**Description**
7	x	Parallel Port D pin 7 is always a parallel port I/O pin.
6	0	Parallel Port D pin 6 is a parallel port I/O pin.
	1	Parallel Port D pin 6 drives ATxA.
5	x	Parallel Port D pin 5 is always a parallel port I/O pin.
4	0	Parallel Port D pin 4 is a parallel port I/O pin.
	1	Parallel Port D pin 4 drives ATxB.
3:0	x	Parallel Port D pins 3:0 are always parallel port I/O pins.

18. SERIAL PORTS E – F

18.1 Overview

Serial Ports E and F are identical to each other, and their asynchronous operation is identical to that of Serial Ports A – D except for the source of the data clock, the buffer sizes, and the transmit, receive, and clock pins. Each serial port can be used in the asynchronous or the HDLC mode with an internal or external clock.

In the asynchronous mode, either 7 or 8 data bits can be transferred, and both a parity bit and/or an additional address (0) or long stop (1) bit can be appended as well. Parity and the address/long stop bits are also detected when they are received. The asynchronous mode is full-duplex.

The transmit and receive buffers of Serial Ports E and F have 4 bytes each; this reduces the interrupt overhead requirements. A serial port interrupt is generated whenever at least one byte is available in the receive buffer or whenever a byte is shifted out of the transmit buffer. The byte is available in the buffer after the final bit is sampled.

The status of each serial port is available in the Serial Port Status Registers (SxSR), and contains information on whether a received byte is available, the receive buffer was overrun, a parity error was received, and the transmit buffer is empty or busy sending a byte. The status is updated when the final bit of a received byte is sampled, or when the final bit of a transmitted byte is sent out.

Serial Ports E and F support the HDLC mode with either an internal or an external clock; separate pins may be used for the transmit and receive clocks, or the transmit and receive clocks may be combined onto a single pin. The HDLC packet flag encapsulation, flag escapes, and CRC calculation and check are handled automatically by the processor. The serial port can detect end-of-frame, short-frame, and CRC errors. Interrupts are generated by the reception of an end-of-frame, at the end of a transmission of a CRC, by an abort sequence, or by a closing flag. Transmit and receive operations are essentially automatic.

The standard CRC-CCITT polynomial ($x^{16} + x^{12} + x^5 + 1$) is implemented for the CRC, with the generator and checker preset to all ones.

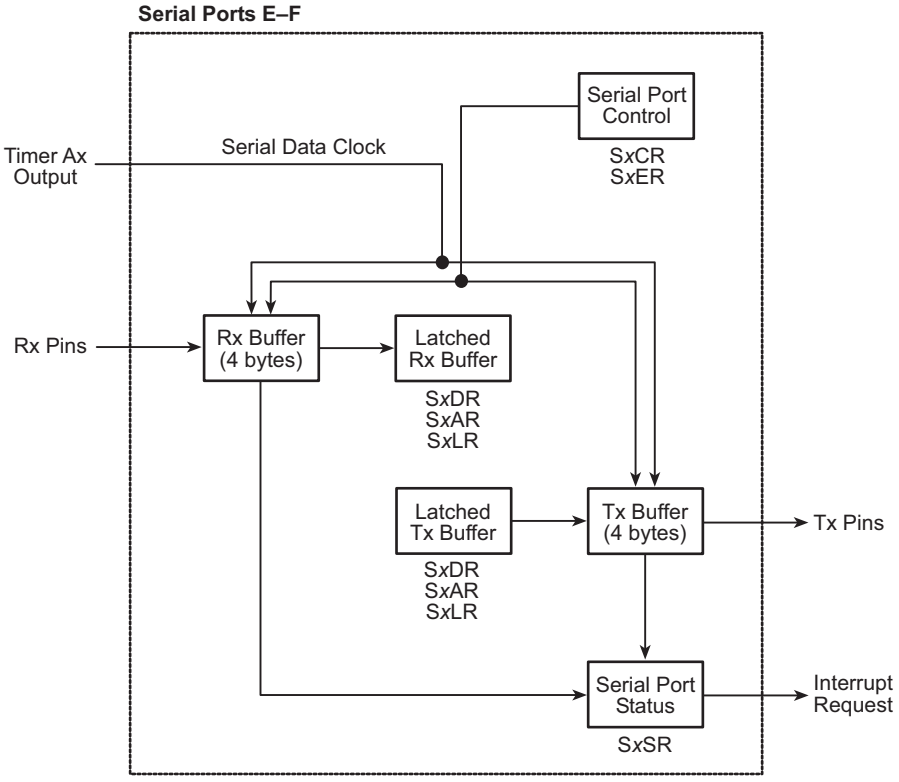
It is possible to send packets with or without a CRC appended. It is also possible to select whether an abort or flag will be transmitted if the transmitter underflows. A packet under transition can be aborted and the abort pattern sent. The idle condition of the line can be flags or all ones.

Several types of data encoding are available in HDLC mode: NRZ, NRZI, biphase-level (Manchester), biphase-space (FM0), and biphase-mark (FM1). IrDA-compliant RZI encoding is also available in the HDLC mode; it reduces the bit widths to one quarter the normal width, which allows the serial-port signal to be connected directly to an IrDA transceiver.

If an internal clock is selected, the serial port data clocks can be generated from the appropriate 8-bit timer (Timer A2 for Serial Port E and Timer A3 for Serial Port F). In the HDLC mode, the bit data rate is equal to the data clock rate divided by 16.

When using an external clock, a 1× (same speed as the data rate) clock is supported. In this case, the maximum data rate is 1/6 of the peripheral clock rate. The receive clock is generated from the transitions in the data stream via a digital phase-locked loop (DPLL). The timing of this synchronization is adjusted with each incoming transition, allowing for tracking if the two external clocks differ slightly in frequency. For more on the clock synchronization and data encoding, see Section 18.3.3.

18.1.1 Block Diagram



18.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port E Data Register	SEDR	0x00C8	R/W	xxxxxxxx
Serial Port E Address Register	SEAR	0x00C9	W	xxxxxxxx
Serial Port E Long Stop Register	SELR	0x00CA	W	xxxxxxxx
Serial Port E Status Register	SESR	0x00CB	R	0xx00000
Serial Port E Control Register	SECR	0x00CC	W	xx000000
Serial Port E Extended Register	SEER	0x00CD	R/W	000x000x
Serial Port F Data Register	SFDR	0x00D8	R/W	xxxxxxxx
Serial Port F Address Register	SFAR	0x00D9	W	xxxxxxxx
Serial Port F Long Stop Register	SFLR	0x00DA	W	xxxxxxxx
Serial Port F Status Register	SFSR	0x00DB	R	0xx00000
Serial Port F Control Register	SFCR	0x00DC	W	xx000000
Serial Port F Extended Register	SFER	0x00DD	R/W	000x000x

18.2 Dependencies

18.2.1 I/O Pins

Serial Port E can transmit on parallel port pin PG6, and can receive on pins PG7 or PG3. If the HDLC mode is enabled, the transmit serial clock is either transmitted or received on PG4, while the receive serial clock is either transmitted or received on PG5.

Serial Port F can transmit on parallel port pin PG2, and can receive on pins PG3 or PG1. If the HDLC mode is enabled, the transmit serial clock is either transmitted or received on PG0, while the receive serial clock is either transmitted or received on PG1.

Table 18-1. Pin Usage Serial Ports E – F

Function	Serial Port E	Serial Port F
Transmit	PG6	PG2
Receive	PG7, PG5*	PG3, PG1*
Transmit Clock	PG4	PG0
Receive Clock	PG5, PG4*	PG1, PG0*

* Introduced with Rabbit 3000A chip.

18.2.2 Clocks

The data clocks for Serial Ports E – F are based on the corresponding Timer A output and can be divided by a Timer A divider. The overall clock divider will be the value in the appropriate register plus one.

18.2.3 Other Registers

Register	Function
TAT2R	Time constant for Serial Port E
TAT3R	Time constant for Serial Port F
PGFR	Alternate port output selection

18.2.4 Interrupts

In the asynchronous mode, a serial port interrupt can be generated whenever a byte is available in the receive buffer or when a byte is finished being transmitted out of the transmit buffer. In the HDLC mode, interrupts are also generated by the reception of an end-of-frame (with abort, valid CRC, or CRC error), at the end of a transmission of a CRC, by an abort sequence, or by a closing flag.

The serial port interrupt vectors are located in the IIR as follows.

- Serial Port E at offset 0x1C0
- Serial Port F at offset 0x1D0

Each of them can be set as Priority 1, 2, or 3 in SxCR, where x is E – F for the two serial ports.

18.3 Operation

18.3.1 Asynchronous Mode

The steps to set up Serial Ports E – F for asynchronous operation are identical to those described in Section 17.3.1 to set up Serial Ports A – D.

18.3.2 HDLC Mode

The following steps explain how to set up Serial Ports E – F for the HDLC mode. When the internal clock is selected, the Rabbit 3000 is in control of all transmit and receive operations, so an interrupt is not required. When an external clock is selected, operations can be handled by either polling the status byte or by a serial port interrupt; the performance will be better with an interrupt.

1. Write the interrupt vector for the interrupt service routine to the internal interrupt table.
2. Set up the desired data transmit and clock pins by writing to the appropriate parallel port function register (PGFR).
3. Select the appropriate mode by writing to SxCR (receive input port and clock source). Also select the interrupt priority.
4. Select additional options by writing to SxER (data encoding, idle line condition, under-run behavior, and combined or separate clocks).
5. Write the desired divider value to TATxR for the appropriate serial port. The overall clock divider will be the value in the appropriate register plus one.
6. To start transmission of a packet, write the first byte to SxDR. If an internal clock is selected, the transmission will begin immediately; if an external clock is selected the transmission will begin when the clock is detected.
7. Continue writing bytes when space is available in the transmit buffer until the final byte of the packet. If a CRC is to be appended to the packet, write the final byte to SxAR. If no CRC is required, write the final byte to SxLR and just a closing flag will be appended. If it is desirable to abort the current packet, write 11 to bits 6–7 of SxCR, and an abort pattern will be transmitted.
8. The receiver will be synchronized on flag bytes and will reset the CRC. By monitoring the received bytes, decisions can be made about the incoming packet; if it is not desired (i.e., it is not addressed to this device), writing a 01 to bits 6–7 of SxCR will force the receiver back into the flag search mode.

A sample HDLC interrupt handler is shown below for Serial Port E.

```
hdlc_sere_isr::
   push af
   ioi ld a, (SESR)     ; get status
   bit 7,a              ; check if byte ready in RX buffer
   push af              ; save status for next check
   jr  z, check_for_tx
rx_ready:

   ; check status byte in A for abort or invalid CRC flags

   ioi ld a, (SEDR)     ; read byte and clear interrupt

   ; store byte in A here

check_for_tx:
   pop af
   bit 3,a              ; check if TX buffer was emptied
   jr nz, done

   ; check status byte in A for transmit finish reason (CRC, abort, etc.)

   ; get next byte to be transmitted into A here; if it is the last
   ;   byte of the packet, load it into SEAR or SELR instead

   ioi ld (SEDR), a     ; load next byte into buffer and clear interrupt
done:
   pop af
   ipres
   ret
```

18.3.3 More on Clock Synchronization and Data Encoding

The transmitter is not capable of sending an arbitrary number of bits, but only a multiple of bytes. However, the receiver can receive frames of any bit length. If the last "byte" in the frame is not eight bits, the receiver sets a status flag that is buffered along with this last byte. Software can then use the table below to determine the number of valid data bits in this last "byte." Note that the receiver transfers all bits between the opening and closing flags, except for the inserted zeros, to the receiver data buffer.

Last Byte Bit Pattern	Valid Data Bits
bbbbbbb0	7
bbbbbb01	6
bbbbb011	5
bbbb0111	4
bbb01111	3
bb011111	2
b0111111	1

Several types of data encoding are available in the HDLC mode. In addition to the normal NRZ, they are NRZI, biphase-level (Manchester), biphase-space (FM0), and biphase-mark (FM1). Examples of these encodings are shown below. Note that the signal level does not convey information in NRZI, biphase-space, and biphase-mark. Instead it is the placement of the transitions that determine the data. In biphase-level it is the polarity of the transition that determines the data.

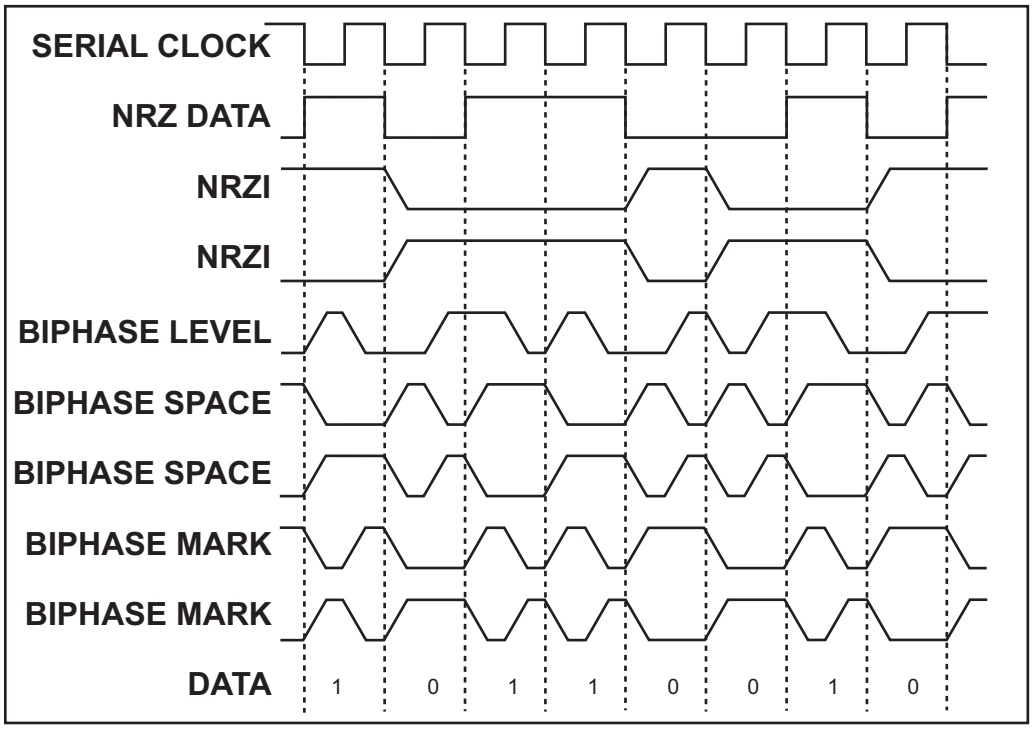


Figure 18-1. Examples of Data Encoding In the HDLC Mode

In the HDLC mode the internal clock comes from the output of Timer A2/Timer A3. The timer output is divided by 16 to form the transmit clock, and is fed to the digital phase-locked loop (DPLL) to form the receive clock. The DPLL is basically just a divide-by-16 counter that uses the timing of the transitions on the receive data stream to adjust its count. The DPLL adjusts the count so that the DPLL output will be properly placed in the bit cells to sample the receive data. To work properly, then, transitions are required in the receive data stream. NRZ data encoding does not guarantee transitions in all cases (a long string of zeros, for example), but the other data encodings do. NRZI guarantees transitions because of the inserted zeros, and the biphase encodings all have at least one transition per bit cell.

The DPLL counter normally counts by 16, but if a transition occurs earlier or later than expected, the count will be modified during the next count cycle. If the transition occurs earlier than expected, it means that the bit cell boundaries are early with respect to the DPLL-tracked bit-cell boundaries, so the count is shortened by either one or two counts. If the transition occurs later than expected, it means that the bit-cell boundaries are late with

respect to the DPLL-tracked bit-cell boundaries, so the count is lengthened by either one or two counts. The decision to adjust by one or by two depends on how far off the DPLL-tracked bit cell boundaries are. This tracking allows for minor differences in the transmit and receive clock frequencies.

With NRZ and NRZI data encoding, the DPLL counter runs continuously, and adjusts after every receive data transition. Since NRZ encoding does not guarantee a minimum density of transitions, the difference between the sending data rate and the DPLL output clock rate must be very small, and depends on the longest possible run of zeros in the received frame. NRZI encoding guarantees at least one transition every six bits (with the inserted zeros). Since the DPLL can adjust by two counts every bit cell, the maximum difference between the sending data rate and the DPLL output clock rate is 1/48 (~2%).

With biphase data encoding (either biphase-level, biphase-mark, or biphase-space), the DPLL runs only as long as transitions are present in the receive data stream. Two consecutive missed transitions causes the DPLL to halt operation and wait for the next available transition. This mode of operation is necessary because it is possible for the DPLL to lock onto the optional transitions in the receive data stream. Since they are optional, they will eventually not be present, and the DPLL can attempt to lock onto the required transitions. Since the DPLL can adjust by one count every bit cell, the maximum difference between the sending data rate and the DPLL output clock rate is 1/16 (~6%).

With biphase data encoding, the DPLL is designed to work in multiple-access conditions where there might not be flags on an idle line. The DPLL will generate an output clock correctly based on the first transition in the leading zero of an opening flag. Similarly, only the completion of the closing flag is necessary for the DPLL to provide the extra two clocks to the receiver to assemble the data correctly. The transition is specified as follows.

- In the biphase-level mode this means the transition that defines the last zero of the closing flag.
- In the biphase-mark and the biphase-space modes this means the transition that defines the end of the last zero of the closing flag.

Figure 18-2 shows the adjustment ranges and output clock for the different modes of operation of the DPLL. Each mode of operation will be described in turn.

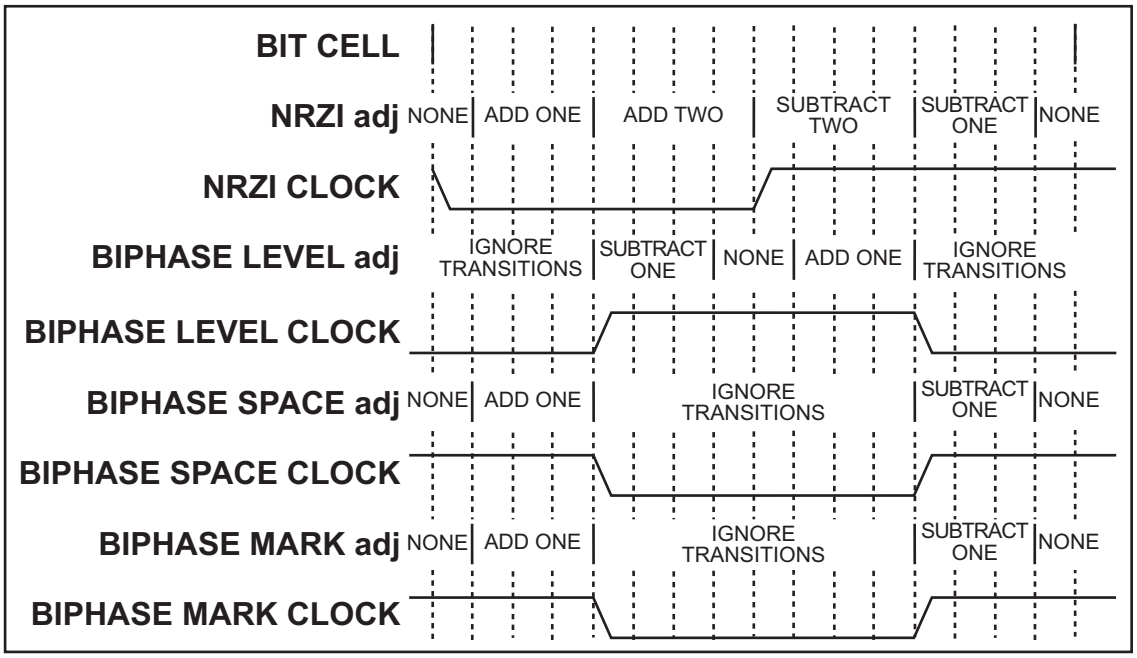


Figure 18-2. Adjustment Ranges and Output Clock for Different DPLL Modes

With NRZ and NRZI encoding, all transitions occur on bit-cell boundaries and the data should be sampled in the middle of the bit cell. If a transition occurs after the expected bit-cell boundary (but before the midpoint), the DPLL needs to lengthen the count to line up the bit-cell boundaries. This corresponds to the "add one" and "add two" regions shown. If a transition occurs before the bit-cell boundary (but after the midpoint), the DPLL needs to shorten the count to line up the bit-cell boundaries. This corresponds to the "subtract one" and "subtract two" regions shown. The DPLL makes no adjustment if the bit-cell boundaries are lined up within one count of the divide-by-16 counter. The regions that adjust the count by two allow the DPLL to synchronize faster to the data stream when starting up.

With biphase-level encoding, there is a guaranteed "clock" transition at the center of every bit cell and optional "data" transitions occur at the bit cell boundaries. The DPLL only uses the clock transitions to track the bit-cell boundaries by ignoring all transitions occurring outside a window around the center of the bit cell. This window is half a bit cell wide. Additionally, because the clock transitions are guaranteed, the DPLL requires that they always be present. If no transition is found in the window around the center of the bit cell for two successive bit cells, the DPLL is not in lock and immediately enters the search mode. The search mode assumes that the next transition seen is a clock transition and immediately synchronizes to this transition. No clock output is provided to the receiver during the search operation. Decoding biphase-level data requires that the data be sampled at either the quarter or three-quarter point in the bit cell. The DPLL here uses the quarter point to sample the data.

Biphase-mark encoding and biphase-space encoding are identical as far as the DPLL is concerned, and are similar to biphase-level encoding. The primary difference is the placement of the clock and data transitions. With these encodings the clock transitions are at the bit-cell boundary, the data transitions are at the center of the bit cell, and the DPLL operation is adjusted accordingly. Decoding biphase-mark or biphase-space encoding requires that the data be sampled by both edges of the recovered receive clock.

18.4 Register Descriptions

Serial Port x Data Register	(SEDR) (SFDR)	(Address = 0x00C8) (Address = 0x00D8)
Bit(s)	**Value**	**Description**
7:0	Read	Returns the contents of the receive buffer.
	Write	Loads the transmit buffer with a data byte for transmission.

Serial Port x Address Register	(SEAR) (SFAR)	(Address = 0x00C9) (Address = 0x00D9)
Bit(s)	**Value**	**Description**
7:0	Read	Returns the contents of the receive buffer.
	Write	Loads the transmit buffer with an address byte, marked with a “zero” address bit, for transmission. In the HDLC mode, the last byte of a frame must be written to this register to enable subsequent CRC and closing flag transmission.

Serial Port x Long Stop Register	(SELR) (SFLR)	(Address = 0x00CA) (Address = 0x00DA)
Bit(s)	**Value**	**Description**
7:0	Read	Returns the contents of the receive buffer.
	Write	Loads the transmit buffer with an address byte, marked with a “one” address bit, for transmission. In the HDLC mode, the last byte of a frame is written to this register to enable subsequent closing flag transmission.

Serial Port x Status Register (Asynchronous Mode Only) (SESR) (SFSR) (Address = 0x00CB) (Address = 0x00DB)		
Bit(s)	**Value**	**Description**
7	0	The receive data register is empty—no input character is ready.
	1	There is a byte in the receive buffer. The transition from "0" to "1" sets the receiver interrupt request flip-flop. The interrupt FF is cleared when the character is read from the data buffer. The interrupt FF will be immediately set again if there are more characters available in the FIFO or shift register to be transferred into the data buffer.
6	0	The byte in the receive buffer is data, received with a valid stop bit.
	1	The address bit or 9th (8th) bit received. This bit is set if the character in the receiver data register has a 9th (8th) bit. This bit is cleared and should be checked before reading a data register since a new data value with a new address bit may be loaded immediately when the data register is read. The byte in the receive buffer is an address, or a byte with a framing error. If an address bit is not expected. If the data in the buffer is all zeros, this may be a break.
5	0	The receive buffer was not overrun.
	1	The receive buffer was overrun. This bit is cleared by reading the receive buffer.
4	0	This bit is always zero in the asynchronous mode
3	0	The transmit buffer is empty.
	1	The transmit data buffer is full. This bit is set when the transmit data register is full, that is, a byte is written to the serial port data register. It is cleared when a byte is transferred to the transmitter shift register or FIFO, or a write operation is performed to the serial port status register. This bit will request an interrupt on the transition from 1 to 0 if interrupts are enabled. Transmit interrupts are cleared when the transmit buffer is written, or any value (which will be ignored) is written to this register.
2	0	The transmitter is idle.
	1	Transmit busy bit. This bit is set if the transmit shift register is busy sending data. It is set on the falling edge of the start bit, which is also the clock edge that transfers data from the transmit data register to the transmit shift register. The transmit busy bit is cleared at the end of the stop bit of the character sent. This bit will cause an interrupt to be latched when it goes from busy to not busy status after the last character has been sent (there are no more data in the transmit data register).
1:0	00	These bits are always zero in the asynchronous mode.

Serial Port x Status Register (HDLC Mode Only)		(SESR) (SFSR)	(Address = 0x00CB) (Address = 0x00DB)
Bit(s)	**Value**	**Description**	
7	0	The receive data register is empty	
	1	There is a byte in the receive buffer. The serial port will request an interrupt while this bit is set. The interrupt is cleared when the receive buffer is empty.	
6,4	00	The byte in the receive buffer is data.	
	01	The byte in the receive buffer was followed by an abort.	
	10	The byte in the receive buffer is the last in the frame, with valid CRC.	
	11	The byte in the receive buffer is the last in the frame, with a CRC error.	
5	0	The receive buffer was not overrun.	
	1	The receive buffer was overrun. This bit is cleared by reading the receive buffer.	
3	0	The transmit buffer is empty.	
	1	The transmit buffer is not empty. The serial port will request an interrupt when the transmitter takes a byte from the transmit buffer, unless the byte is marked as the last in the frame. Transmit interrupts are cleared when the transmit buffer is written, or when any value (which will be ignored) is written to this register.	
2:1	00	Transmit interrupt due to buffer empty condition.	
	01	Transmitter finished sending CRC. An interrupt is generated at the end of the CRC transmission. Data written in response to this interrupt will cause only one flag to be transmitted between frames, and no interrupt will be generated by this flag.	
	10	Transmitter finished sending an abort. An interrupt is generated at the end of an abort transmission.	
	11	The transmitter finished sending a closing flag. Data written in response to this interrupt will cause at least two flags to be transmitted between frames.	
0	0	The byte in the receiver buffer is 8 bits.	
	1	The byte in the receiver buffer is less than 8 bits.	

Serial Port x Control Register		(SECR) (Address = 0x00CC) (SFCR) (Address = 0x00DC)
Bit(s)	**Value**	**Description**
7:6	00	No operation. These bits are ignored in the asynchronous mode.
	01	In the HDLC mode, force receiver in flag search mode.
	10	No operation.
	11	In the HDLC mode, transmit an abort pattern.
5	0	Enable the receiver input.
	1	Disable the receiver input.
4	0	Pin PG7 (Serial Port E) or PG3 (Serial Port F) is used for the receiver input.
	1*	Pin PG5 (Serial Port E) or PG1 (Serial Port F) is used for the receiver input.
3:2	00	Asynchronous mode with 8 bits per character.
	01	Asynchronous mode with 7 bits per character. In this mode the most significant bit of a byte is ignored for transmit, and is always zero in receive data.
	10	HDLC mode with external clock. The external clocks are supplied as follows: • Bit 4 is 0—pins PG4 and PG5(Serial Port E); pins PG0 and PG1 (Serial Port F). • Bit 4 is 1—pin PG4 (Serial Port E) and pin PG0 (Serial Port F).
	11	HDLC mode with internal clock. The clock is 16× the data rate, and the DPLL is used to recover the receive clock. If necessary, the clocks are supplied as follows: • Bit 4 is 0—pins PG4 and PG5(Serial Port E); pins PG0 and PG1 (Serial Port F). • Bit 4 is 1—pin PG4 (Serial Port E) and pin PG0 (Serial Port F).
1:0	00	The serial port interrupt is disabled.
	01	The serial port uses Interrupt Priority 1.
	10	The serial port uses Interrupt Priority 2.
	11	The serial port uses Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

Serial Port x Extended Register (Asynchronous Mode Only)		(SEER) (SFER)	(Address = 0x00CD) (Address = 0x00DD
Bit(s)	**Value**	**Description**	
7:5	xxx	These bits are ignored in the asynchronous mode and should be set to zeros.	
4	0	Normal asynchronous data encoding.	
	1	Enable RZI coding (3/16 bit cell IrDA-compliant).	
3	0	Normal break operation. This option should be selected when address bits are expected.	
	1	Fast break termination. At the end of break, a dummy character is written to the buffer, and the receiver can start character assembly after one bit time.	
2	0	Asynchronous clock is 16× data rate.	
	1	Asynchronous clock is 8× data rate.	
1:0	xx	These bits are ignored in the asynchronous mode.	

Serial Port x Extended Register (HDLC Mode Only)		(SEER) (SFER)	(Address = 0x00CD) (Address = 0x00DD)
Bit(s)	**Value**	**Description**	
7:5	000	NRZ data encoding for HDLC receiver and transmitter.	
	010	NRZI data encoding for HDLC receiver and transmitter.	
	100	Biphase-level (Manchester) data encoding for HDLC receiver and transmitter.	
	110	Biphase-space data encoding for HDLC receiver and transmitter.	
	111	Biphase-mark data encoding for HDLC receiver and transmitter.	
4	0	Normal HDLC data encoding.	
	1	Enable RZI coding (¼ bit cell IrDA-compliant). This mode can only be used with an internal clock and NRZ data encoding.	
3	0	Idle line condition is flags.	
	1	Idle line condition is all ones.	
2	0	Transmit flag on underrun.	
	1	Transmit abort on underrun.	
1:0	xx	These bits are ignored in the HDLC mode and are set to zeros.	

Timer A Time Constant x Register		(TAT2R) (TAT3R)	(Address = 0x00A5) (Address = 0x00A7))
Bit(s)	**Value**	**Description**	
7:0		Time constant for the Timer A counter. This time constant will take effect the next time that the Timer A counter counts down to zero. The timer counts modulo $n + 1$, where n is the programmed time constant.	

Parallel Port G Function Register (PGFR) (Address = 0x004D)		
Bit(s)	**Value**	**Description**
7	0	Parallel Port G pin 7 is always a parallel port input.
	1	Parallel Port G pin 7 is enabled as RxE.
6	0	Parallel Port G pin 6 is a parallel port output.
	1	Parallel Port G pin 6 drives TxE.
5	0	Parallel Port G pin 5 is always a parallel port input.
	1	Parallel Port G pin 5 is enabled as an SDLC/HDLC.receive clock output for Serial Port E.
4	0	Parallel Port G pin 4 is a parallel port output.
	1	Parallel Port G pin 4 is enabled as an SDLC/HDLC.transmit clock output for Serial Port E.
3	0	Parallel Port G pin 3 is always a parallel port input.
	1	Parallel Port G pin 3 is enabled as RxF.
2	0	Parallel Port G pin 2 is a parallel port output.
	1	Parallel Port G pin 2 drives TxF.
1	0	Parallel Port G pin 1 is always a parallel port input.
	1	Parallel Port G pin 1 is enabled as an SDLC/HDLC.receive clock output for Serial Port F.
0	0	Parallel Port G pin 0 is a parallel port output.
	1	Parallel Port G pin 0 is enabled as an SDLC/HDLC.transmit clock output for Serial Port F.

19. SLAVE PORT

19.1 Overview

The slave port is a parallel communication port that can be used to communicate with an external master device. The slave port consists of three data input and data output registers, and a status register.

The data input registers are written by the master (the external device) and are read by the processor. The data output registers are written by the processor and are read by the master. Note that the data registers are named from the point of view of the processor. The slave device can only read the data input registers and write to the data output registers. Similarly, the master device can only read the data input registers and write the data output registers. Both devices can read and write to the status register.

The status register contains the interrupt status bits and a status flag corresponding to each data input or data output register to indicate the empty or full status of the data register. Data registers are marked full when written by the source side of the interface, and are marked empty when read by the destination side of the interface.

The hardware interface to the external master consists of an 8-bit bidirectional data bus with a read strobe, write strobe, and chip select. There are two address lines that select one of the three data registers or the status register.

Table 19-1. Slave Port Addresses

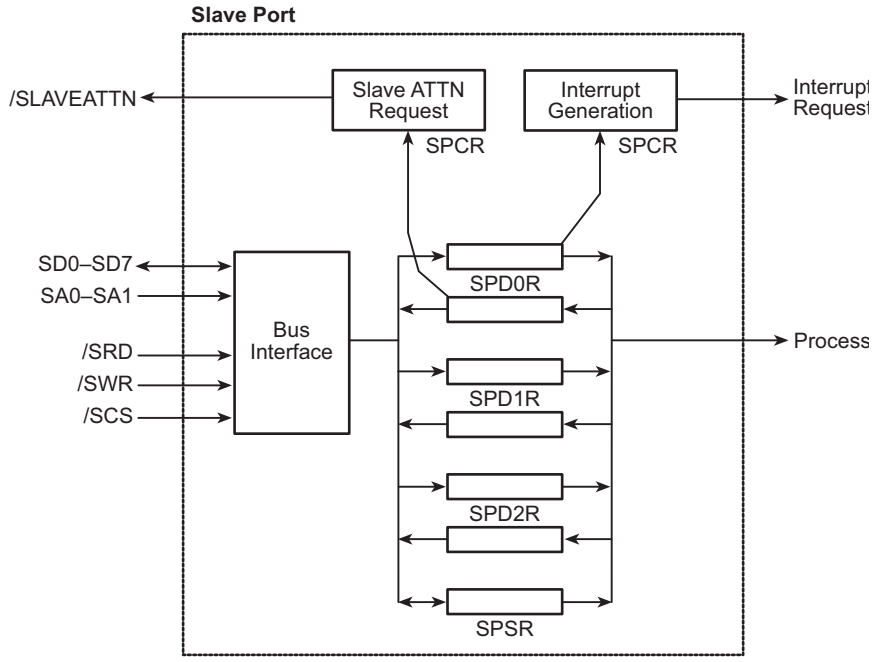
Slave Port Address	Slave Port Register
00	Data Register 0
01	Data Register 1
10	Data Register 2
11	Status Register

A slave attention signal is asserted when the processor writes to one of the slave port data registers (SPD0R), and can be deasserted by the master by performing a dummy write to the status register. This signal can be used to interrupt the master to indicate that the master needs to read data from the slave.

The slave port interrupt is asserted when the master writes to SPD0R. The processor clears this interrupt condition by writing to the status register.

The slave port can be used to bootstrap the processor by setting the SMODE pins appropriately. See Chapter 3 for more information on this mode.

19.1.1 Block Diagram



19.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Slave Port Data 0 Register	SPD0R	0x0020	R/W	xxxxxxxx
Slave Port Data 1 Register	SPD1R	0x0021	R/W	xxxxxxxx
Slave Port Data 2 Register	SPD2R	0x0022	R/W	xxxxxxxx
Slave Port Status Register	SPSR	0x0023	R	00000000
Slave Port Control Register	SPCR	0x0024	R/W	0xx00000

19.2 Dependencies

19.2.1 I/O Pins

When the slave port is enabled by writing to SPCR, the following pins are enabled for slave port mode. Note that enabling the slave port mode will override any general-purpose I/O or external I/O bus settings for these pins; when the slave port is enabled they will perform slave port functionality.

Table 19-2. Slave Port Pin Functionality

Pin(s)	Slave Port Signal	Direction	Functionality
PA0–PA7	SD0–SD7	Bidirectional	Slave data bus
PB7	/SLAVEATTN	Output	Slave interrupt request (output)
PB6	/ASCS*	Input	Alternate slave chip select
PB4–PB5	SA0–SA1	Input	Slave address bus
PB3	/SRD	Input	Slave port read strobe
PB2	/SWR	Input	Slave port write strobe
PE7	/SCS	Input	Slave chip select

* Introduced with Rabbit 3000A chip

19.2.2 Clocks

All slave port operations are based on the processor clock.

19.2.3 Interrupts

If slave port interrupts are enabled, a slave port interrupt will occur on the slave device whenever the master writes to SPD0R. The /SLAVEATTN pin is asserted whenever the slave device writes to SPD0R. Either if these conditions is cleared when either the master or slave reads or writes any of the slave port registers.

The slave port interrupt vector is in the IIR at offset 0x080. It can be set as Priority 1, 2, or 3 by writing to SPCR.

19.3 Operation

Figure 19-1 shows a typical slave port connection between a Rabbit processor as the master and two slaves.

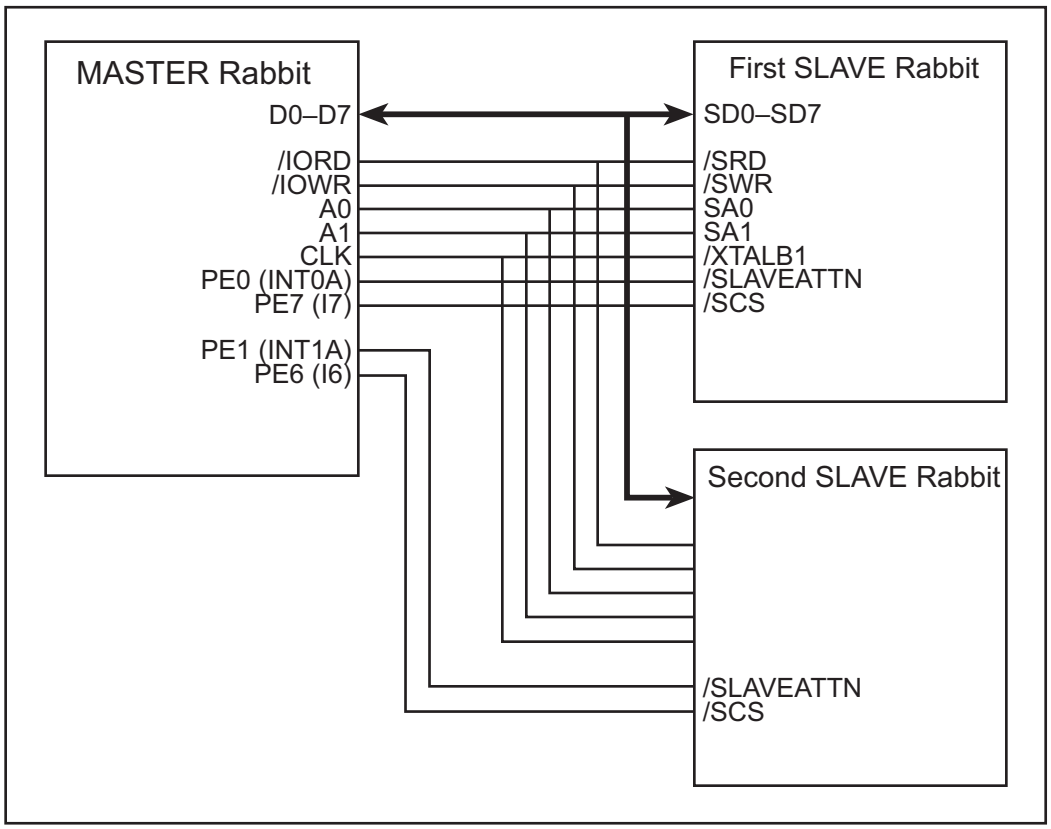


Figure 19-1. Master/Slave Port Connections

These connections are summarized in Table 19-3.

Table 19-3. Typical Slave Port Connections

Master		Slave #1		Slave #2	
Data Bus	D0–D7	SD0–SD7	PA0–PA7	SD0–SD7	PA0–PA7
Address Bus	A0–A1	SA0–SA1	PB4–PB5	SA0–SA1	PB4–PB5
I/O Read Strobe	/IORD	/SRD	PB3	/SRD	PB3
I/O Write Strobe	/IOWR	/SWR	PB2	/SWR	PB2
Slave #1 Chip Select (I/O strobe I7)	PE7	/SCS	PB6	–	—
Slave #2 Chip Select (I/O strobe I6)	PE6	—	—	/SCS	PB6
External Interrupt 0 (from Slave #1)	PE0	/SLAVEATTN	PB7	—	—
External Interrupt 1 (from Slave #2)	PE1	—	—	/SLAVEATTN	PB7

Note that the slave port on the master Rabbit processor is *not* used; the master uses the data bus to send and receive data to the slave port data registers on the slave devices.

In this setup, pins PE6 and PE7 are set up as I/O strobe chip selects for the two slave devices, and PE0 and PE1 are used as external interrupt inputs to monitor the /SLAVEATTN signals from the slaves.

In this setup, the slave port is used as follows:

- The slave responds to the interrupt and reads the slave port data registers.
- When the slave wishes to send data to the master, it writes the slave port data registers, writing SPD0R last, which enables the /SLAVEATTN signal.
- When the master detects the change on /SLAVEATTN, it reads the slave port data registers.

19.3.1 Master Setup

1. Enable the I/O strobes on PE6 and PE7 by writing to the appropriate Parallel Port E pin and external I/O registers.
2. Enable the external interrupts on PE0 and PE1 by writing to the appropriate external interrupt registers.

19.3.2 Slave Setup

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Configure SPCR to select the interrupt priority (note that interrupts will be enabled once this value is set).

19.3.3 Master/Slave Communication

1. The master writes data to the appropriate external I/O address on the data bus for the slave device and register desired. For example, in the setup described here, the master would write to register SPD2R on the first slave by writing to the address 0xC002 (0xC000 for the I6 strobe, and 0x0002 for SPD2R on that slave).
2. If the master is writing multiple bytes, it should write to SPD0R last since that will trigger an interrupt on the slave device. If only one byte is being sent, it should be written to SPD0R.
3. The slave responds to the interrupt, reading the data from the slave port data registers.

19.3.4 Slave/Master Communication

1. The slave writes data to the appropriate slave port data register. If it is writing multiple bytes, SPD0R should be written last, which enables the /SLAVEATTN line.
2. The master receives an external interrupt from the /SLAVEATTN line, and reads the data out of the slave port data registers via external I/O reads on the data bus.

19.3.5 Handling Interrupts

The interrupt request on the slave is cleared by either the master or the slave accessing one of the slave port registers. To clear the interrupt without affecting the register values, a dummy write can be made to SPSR.

19.3.6 Example ISR

A sample interrupt handler is shown below.

```
slave_isr::
   push af             ; save used registers

   ; read the data sent by the master
   ioi ld a, (SPD2R)
   ld (to_slv_d2), a
   ioi ld a, (SPD1R)
   ld (to_slv_d1), a
   ioi ld a, (SPD0R)
   ld (to_slv_d0), a

   ; if a response is required, perform it here
   ld a, (to_mas_d2)
   ioi ld (SPD2R), a
   ld a, (to_mas_d1)
   ioi ld (SPD1R), a
   ld a, (to_mas_d0)
   ioi ld (SPD0R), a   ; this write asserts /SLVATTN

   ; the interrupt request is cleared by any read/write of the registers

   pop af              ; restore used registers
   ipres
   ret
```

19.3.7 Other Configurations

There are other slave port configurations possible:

- The master could use the external I/O bus instead of the memory bus.
- All devices could poll the slave port status register to determine when data is present instead of relying on interrupts.
- The master could write to SPD0R, triggering an interrupt on the slave. The slave could then simply write a response into SPD0R, which the master detects by polling SPSR. This configuration is useful when fewer signals are desired, or the master device has no external interrupts available.

If polling is to be used, it is important to note that not all bits in the status register may be updated at once; it is possible to read a transitional value as the register updates. To guarantee a proper polling read, the status register should be read twice; when the same value is read both times the value is correct.

Similarly, it is possible to receive a scrambled value from a data register if it is read while being written. The protocol used should take account of this and prevent it from occurring (the protocol described above guarantees this will not occur).

19.3.8 Timing Diagrams

Figure 19-2 shows the sequence of events when the master reads/writes the slave port registers.

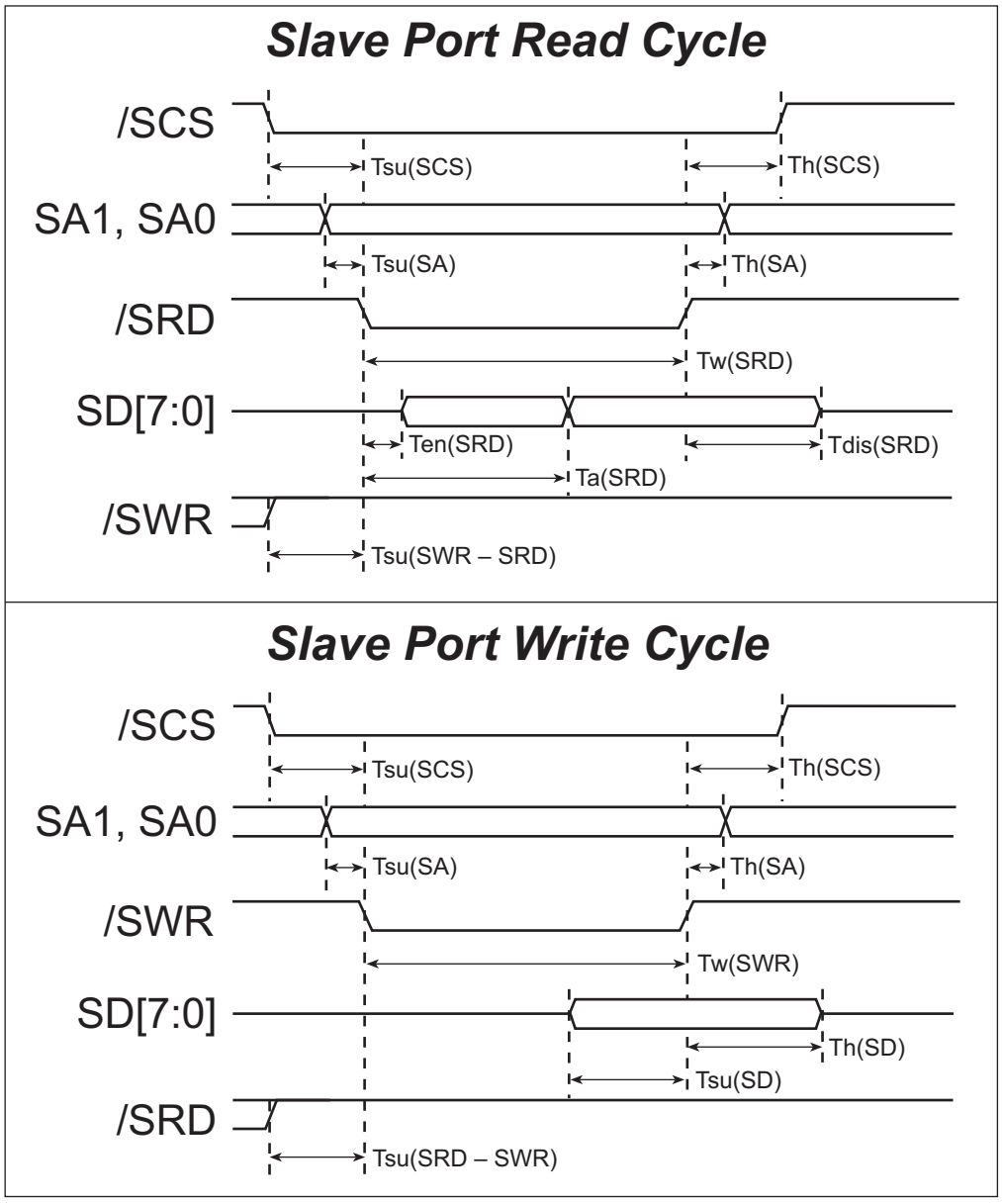


Figure 19-2. Slave Port R/W Timing Diagram

The following table explains the parameters used in Figure 19-2.

Symbol	Parameter	Minimum (ns)	Maximum (ns)
Tsu(SCS)	/SCS Setup Time	5	—
Th(SCS)	/SCS Hold Time	0	—
Tsu(SA)	SA Setup Time	5	—
Th(SA)	SA Hold Time	0	—
Tw(SRD)	/SRD Low Pulse Width	40	—
Ten(SRD)	/SRD to SD Enable Time	0	—
Ta(SRD)	/SRD to SD Access Time	—	30
Tdis(SRD)	/SRD to SD Disable Time	—	15
Tsu(SRW – SRD)	/SWR High to /SRD Low Setup Time	40	—
Tw(SWR)	/SWR Low Pulse Width	40	—
Tsu(SD)	SD Setup Time	10	—
Th(SD)	SD Hold Time	5	—
Tsu(SRD – SWR)	/SRD High to /SWR Low Setup Time	40	—

19.4 Register Descriptions

Slave Port Data x Registers		(SPD0R) (Address = 0x0020) (SPD1R) (Address = 0x0021) (SPD2R) (Address = 0x0022)
Bit(s)	**Value**	**Description**
7:0	Read	The corresponding byte of the slave port is read.
	Write	The corresponding byte of the slave port is written.

Slave Port Status Register		(SPSR) (Address = 0x0023)
Bit(s)	**Value**	**Description**
7	0	Processor wrote to SPSR.
	1	Master wrote to SPD0R.
6	0	Slave port read byte 2 is empty.
	1	Slave port read byte 2 is full.
5	0	Slave port read byte 1 is empty.
	1	Slave port read byte 1 is full.
4	0	Slave port read byte 0 is empty.
	1	Slave port read byte 0 is full.
3	0	Master wrote to SPSR.
	1	Processor wrote to SPD0R.
2	0	Slave port write byte 2 is empty.
	1	Slave port write byte 2 is full.
1	0	Slave port write byte 1 is empty.
	1	Slave port write byte 1 is full.
0	0	Slave port write byte 0 is empty.
	1	Slave port write byte 0 is full.

Slave Port Control Register (SPCR) (Address = 0x0024)		
Bit(s)	**Value**	**Description**
7	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
	Write	These bits are ignored and should be written with zero.
4	0	/SCS from PE7.
	1	/SCS from PB6*.
3:2 (Write only)	00	Disable the slave port. Parallel Port A is a byte-wide input port.
	01	Disable the slave port. Parallel Port A is a byte-wide output port.
	10	Enable the slave port.
	11	Enable the external I/O bus. Parallel Port A is used for the data bus and Parallel Port B[7:2] is used for the address bus.
1:0 (Write only)	00	Slave port interrupts are disabled.
	01	Slave port interrupts use Interrupt Priority 1.
	10	Slave port interrupts use Interrupt Priority 2.
	11	Slave port interrupts use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

20. INPUT CAPTURE

20.1 Overview

The Input Capture peripheral consists of two channels, each of which contains a 16-bit counter and edge-detection circuitry. The Input Capture channels are usually used to determine the time between events. An event is signaled by a rising or falling edge (or optionally by either edge) on one of 16 input pins that can be selected as the input for either of the two channels. The Input Capture channels synchronize their inputs to the Input Capture clock (from Timer A8), providing a low-pass filter functionality on the inputs, as shown in Section 20.2.4.

When capturing an input, the channel starts/stops the counter (clocked by Timer A8) according to the signal edges on various parallel port pins, providing the ability to measure pulse widths and time intervals between external events, time-stamp signal changes on a pin, and measure time intervals between a software start and an external event. An interrupt can also be generated when an edge is detected.

A 16 bit counter is used to record the time at which the event takes place. The counter is driven by the output of Timer A8 and can be set to count at a rate ranging from the full clock speed (perclk/2) down to 1/256 the clock speed (perclk/512).

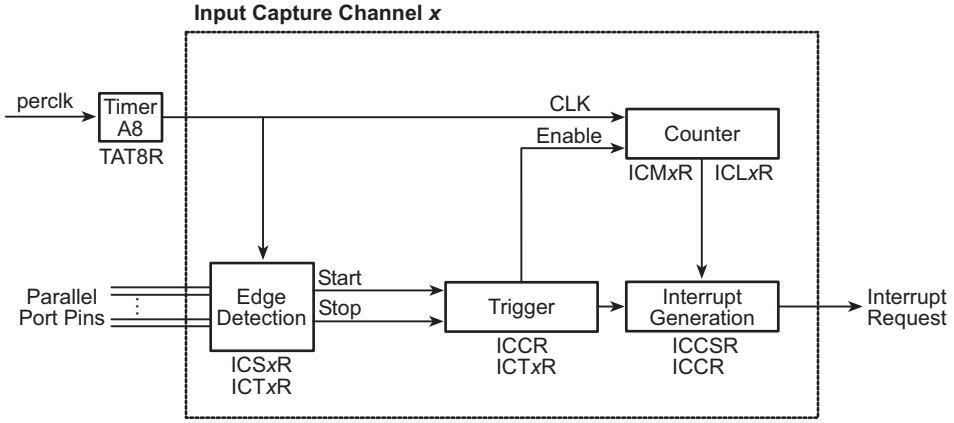
Two events are recognized: a *start condition* and a *stop condition*. The start condition may be used to start counting and the stop condition to stop counting. However, the counter may also run continuously or run until a stop condition is encountered. The start and stop conditions may also be used to latch the current count at the instant the condition occurs rather than actually start or stop the counter. The same pin may be used to detect the start and stop condition—for example a rising edge could be the start condition and a falling edge could be the stop condition. The start and stop condition can also be input on separate pins.

The Input Capture channels can be used to measure the width of fast pulses. This is done by starting the counter on the first edge of the pulse and capturing the counter value on the second edge of the pulse. In this case the maximum error in the measurement is approximately 2 periods of the clock used to count the counter. If there is sufficient time between events for an interrupt to take place the unit can be set up to capture the counter value on either start or stop conditions (or both) and cause an interrupt each time the count is captured. The counter can also be cleared and started under software control and then have its value captured in response to an input.

The capture counter can be synchronized with Timer B outputs to load parallel port output registers. This makes it possible to generate an output signal precisely synchronized with

an input signal. Usually it will be desired to synchronize one of the Input Capture counters with the Timer B counter. The count offset can be measured by outputting a pulse at a precise time using Timer B to set the output time and capturing the output pulse with an Input Capture channel. Once the phase relationship is known between the counters it is then possible to output pulses a precise time delay after an input pulse is captured, provided that the time delay is great enough for the interrupt routine to processes the capture event and set up the output pulse synchronized by Timer B. The minimum time delay needed is probably less than 10 µs if the software is done carefully and the clock speed is reasonably high.

20.1.1 Block Diagram



20.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Input Capture Ctrl/Status Register	ICCSR	0x0056	R/W	00000000
Input Capture Control Register	ICCR	0x0057	W	xxxxxx00
Input Capture Trigger 1 Register	ICT1R	0x0058	W	00000000
Input Capture Source 1 Register	ICS1R	0x0059	W	xxxxxxxx
Input Capture LSB 1 Register	ICL1R	0x005A	R	xxxxxxxx
Input Capture MSB 1 Register	ICM1R	0x005B	R	xxxxxxxx
Input Capture Trigger 2 Register	ICT2R	0x005C	W	00000000
Input Capture Source 2 Register	ICS2R	0x005D	W	xxxxxxxx
Input Capture LSB 2 Register	ICL2R	0x005E	R	xxxxxxxx
Input Capture MSB 2 Register	ICM2R	0x005F	R	xxxxxxxx

20.2 Dependencies

20.2.1 I/O Pins

Each Input Capture channel can accept input from one of the following parallel port pins: PC1, PC3, PC5, PC7, PD1, PD3, PD5, PD7, PF1, PF3, PF5, PF7, PG1, PG3, PG5, PG7. Use ICTxR to select which input pin to trigger on.

Note that these pins can be used for other peripherals at the same time as the Input Capture peripheral. For example, you can use Input Capture to use measure the pulse width on a serial port input to measure the baud rate.

20.2.2 Clocks

The 16-bit Input Capture counters are clocked from the output of Timer A8, and can run at rates from perclk/2 down to perclk/512 by writing the appropriate value to TAT8R.

20.2.3 Other Registers

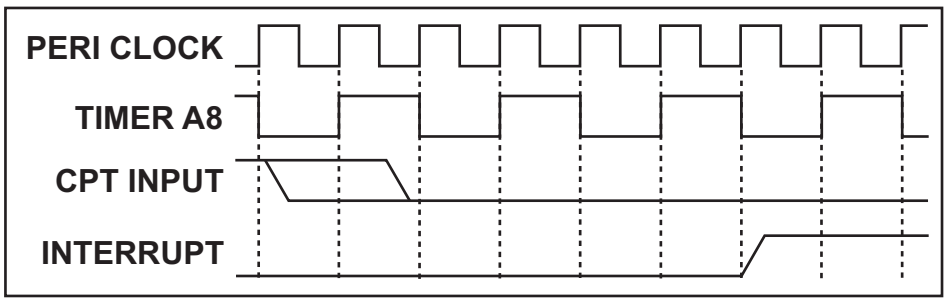
Register	Function
TAT8R	Time constant for Input Capture clock.

20.2.4 Interrupts

Each input capture channel can generate an interrupt whenever a start/stop condition occurs. The interrupt request is cleared when ICCSR is read.

The Input Capture interrupt vector is in the IIR at offset 0x1A0. It can be set as Priority 1, 2, or 3.

The Input Capture channels synchronize their inputs to the peripheral clock (further divided by Timer A8). Since the inputs are only sampled in synch with the peripheral clock, any faster state faster changes cannot be detected, which is akin to a digital low-pass filter functionality on the inputs. Because of this, there is some delay between the input transition and when an interrupt is requested, as shown below. The status bits in ICSxR are set coincident with the interrupt request and are reset when read from the ICSxR.



20.3 Operation

20.3.1 Input-Capture Channel

The following steps explain how to set up an Input Capture channel.

1. Configure Timer A8 via TAT8R to provide the desired Input Capture clock.
2. Configure ICTxR to provide the desired start/stop operation and conditions.
3. Configure ICSxR to select the input pins for the start and stop conditions.
4. Reset the counter by writing to ICCSR.

20.3.2 Handling Interrupts

The following steps explain how an interrupt is used.

1. Write the vector to the interrupt service routine to the internal interrupt table
2. Configure the Input Capture Control/Status Register (ICCSR) to select events that will generate an interrupt.
3. Configure the Input Capture Control Register (ICCR) to select the interrupt priority (note that interrupts will be enabled once this value is set; this step should be done last).

The following actions occur within the interrupt service routine.

- If needed, the current counter value can be read from ICLxR and LCMxR (reading from ICLxR latches the value of ICLxR, so ICLxR should always be read first)
- If the counter is expected to roll over, determine if that is why the interrupt occurred by reading the status bits in ICCSR and adjusting any software counters accordingly
- The interrupt request should be cleared by reading from ICCSR

20.3.3 Example ISR

A sample interrupt handler is shown below.

```
ic_isr::
   push af
   ioi ld a, (ICCSR)    ; clear the interrupt request and get status

   ; determine which interrupts have occurred
   ; if rollover, perform any necessary software counter adjustments here
   ; read counter values

   pop af
   ipres
   ret
```

20.3.4 Example Applications

Pulse Width or Time Between Events

The following steps explain how to measure the pulse width or time between events.

1. Select the same input pin to perform a pulse-width measurement between the start and stop conditions, or select two different input pins to measure time between events on those pins.
2. Set the counter to start on the start condition and stop on the stop condition, latch on the stop condition, and generate an interrupt on the stop condition.
3. In the interrupt handler, read out the counter to determine the pulse width or time interval between the two events.

Time-Stamp External Events

The following steps explain how to time-stamp external events.

1. Set the trigger for the desired event type.
2. Set the counter to run continuously, latch on the start (and/or stop) condition, and generate an interrupt on the start (and/or stop) condition
3. In the interrupt handler, read out the counter as an event timestamp.

Measure Time Interval from a Software Start to an External Event

The following steps explain how to measure the time interval between a software start and the occurrence of an external event.

1. Set up the counter to run continuously, latch on the stop condition, and generate an interrupt on the stop condition.
2. Set up the stop condition for the event of interest.
3. Reset the counter via ICCSR at the software start.
4. In the interrupt handler, read the counter as a time duration.

20.4 Register Descriptions

Input Capture Control/Status Register (ICCSR) (Address = 0x0056)		
Bit(s)	**Value**	**Description**
7 (Read)	0	The Input Capture 2 Start condition has not occurred.
	1	The Input Capture 2 Start condition has occurred.
6 (Read)	0	The Input Capture 2 Stop condition has not occurred.
	1	The Input Capture 2 Stop condition has occurred.
5 (Read)	0	The Input Capture 1 Start condition has not occurred.
	1	The Input Capture 1 Start condition has occurred.
4 (Read)	0	The Input Capture 1 Stop condition has not occurred.
	1	The Input Capture 1 Stop condition has occurred.
3 (Read)	0	The Input Capture 2 counter has not rolled over to all zeros.
	1	The Input Capture 2 counter has rolled over to all zeros.
2 (Read)	0	The Input Capture 1 counter has not rolled over to all zeros.
	1	The Input Capture 1 counter has rolled over to all zeros.
7:2 (Read)		These status bits (but not the interrupt enable bits) are cleared by the read of this register, as is the Input Capture Interrupt.
7:4 (Write)	0	The corresponding Input Capture interrupt is disabled.
	1	The corresponding Input Capture interrupt is enabled.
3 (Write)	0	No effect on Input Capture 2 counter. This bit always reads as zero.
	1	Reset Input Capture 2 counter to all zeros and clears the rollover latch.
2 (Write)	0	No effect on Input Capture 1 counter. This bit always reads as zero.
	1	Reset Input Capture 1 counter to all zeros and clears the rollover latch.
1:0	0x	Normal Input Capture operation.
	x0	Normal Input Capture operation.
	11	Reserved for test. The Input Capture counter increments at both bit 0 and bit 8. There is no carry from lower byte to higher byte.

Input Capture Control Register (ICCR) (Address = 0x0057)		
Bit(s)	**Value**	**Description**
7:2		These bits are ignored.
1:0	00	Input Capture interrupts are disabled.
	01	Input Capture interrupt use Interrupt Priority 1.
	10	Input Capture interrupt use Interrupt Priority 2.
	11	Input Capture interrupt use Interrupt Priority 3.

Input Capture Trigger x Register (ICT1R) (Address = 0x0058) (ICT2R) (Address = 0x005C)		
Bit(s)	**Value**	**Description**
7:6	00	Disable the counter.
	01	The counter runs from the Start condition until the Stop condition.
	10	The counter runs continuously.
	11	The counter runs continuously, until the Stop condition.
5:4	00	Disable the count latching function.
	01	Latch the count on the Stop condition only.
	10	Latch the count on the Start condition only.
	11	Latch the count on either the Start or Stop condition.
3:2	00	Ignore the starting input.
	01	The Start condition is the rising edge of the starting input.
	10	The Start condition is the falling edge of the starting input.
	11	The Start condition is either edge of the starting input.
1:0	00	Ignore the ending input. These two bits are ignored in Counter operation.
	01	The Stop condition is the rising edge of the ending input.
	10	The Stop condition is the falling edge of the ending input.
	11	The Stop condition is either edge of the ending input.

Input Capture Source x Register		(ICS1R) (ICS2R)	(Address = 0x0059) (Address = 0x005D)
Bit(s)	**Value**	**Description**	
7:6	00	Parallel Port C used for Start condition input.	
	01	Parallel Port D used for Start condition input.	
	10	Parallel Port F used for Start condition input.	
	11	Parallel Port G used for Start condition input.	
5:4	00	Use port bit 1 for Start condition input.	
	01	Use port bit 3 for Start condition input.	
	10	Use port bit 5 for Start condition input.	
	11	Use port bit 7 for Start condition input.	
3:2	00	Parallel Port C used for Stop condition input.	
	01	Parallel Port D used for Stop condition input.	
	10	Parallel Port F used for Stop condition input.	
	11	Parallel Port G used for Stop condition input.	
1:0	00	Use port bit 1 for Stop condition input.	
	01	Use port bit 3 for Stop condition input.	
	10	Use port bit 5 for Stop condition input.	
	11	Use port bit 7 for Stop condition input.	

Input Capture LSB x Register		(ICL1R) (ICL2R)	(Address = 0x005A) (Address = 0x005E)
Bit(s)	**Value**	**Description**	
7:0	Read	The least significant eight bits of the latched Input Capture count are returned. Reading the LSB of the count latches the MSB of the count to avoid reading stale data. Reading the MSB of the count opens the latches.	

Input Capture MSB x Register		(ICM1R) (ICM2R)	(Address = 0x005B) (Address = 0x005F)
Bit(s)	**Value**	**Description**	
7:0	Read	The most significant eight bits of the latched Input Capture count are returned.	

Timer A Time Constant 8 Register		(TAT8R)	(Address = 0x00A6)
Bit(s)	**Value**	**Description**	
7:0		Time constant for the Timer A counter. This time constant will take effect the next time that the Timer A counter counts down to zero. The timer counts modulo $n + 1$, where n is the programmed time constant.	

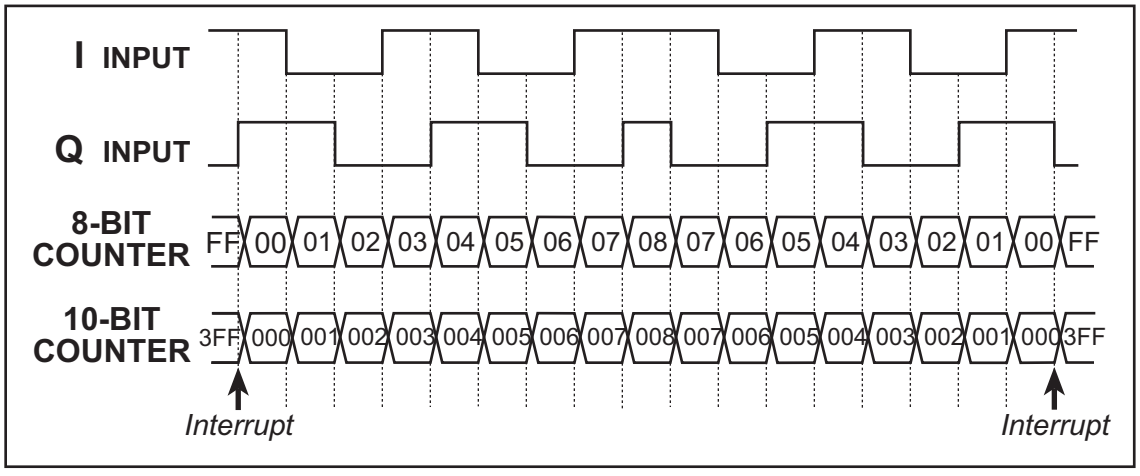
21. QUADRATURE DECODER

21.1 Overview

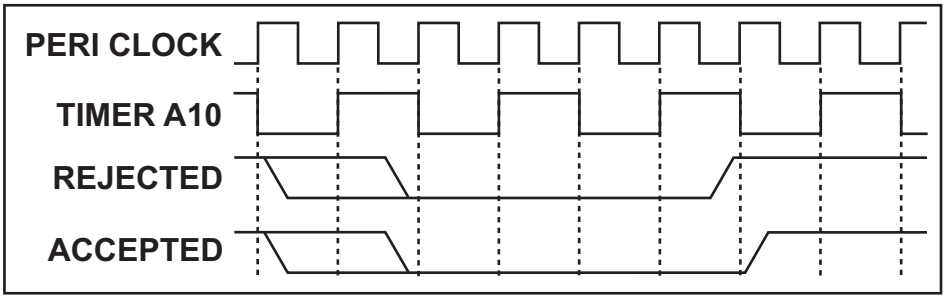
The Rabbit 3000 has a two-channel Quadrature Decoder that accepts inputs via specific pins on Parallel Port F. Each channel has two inputs, the in-phase (I) input and the 90 degree or quadrature-phase (Q) input. An 8 or 10-bit up/down counter counts encoder steps in the forward and backward directions, and provides interrupts when the count goes from 0x00 to 0xFF or from 0xFF to 0x00. An interrupt can occur each time the count overflows or underflows. The Quadrature Decoder contains digital filters on the inputs to prevent false counts. The external signals are synchronized with an internal clock provided by the output of Timer A10.

Each Quadrature Decoder channel accepts inputs from either the upper nibble or lower nibble of Parallel Port F. The I signal is input on an odd-numbered port bit, while the Q signal is input on an even-numbered port bit. There is also a disable selection, which is guaranteed not to generate a count increment or decrement on either entering or exiting the disabled state.

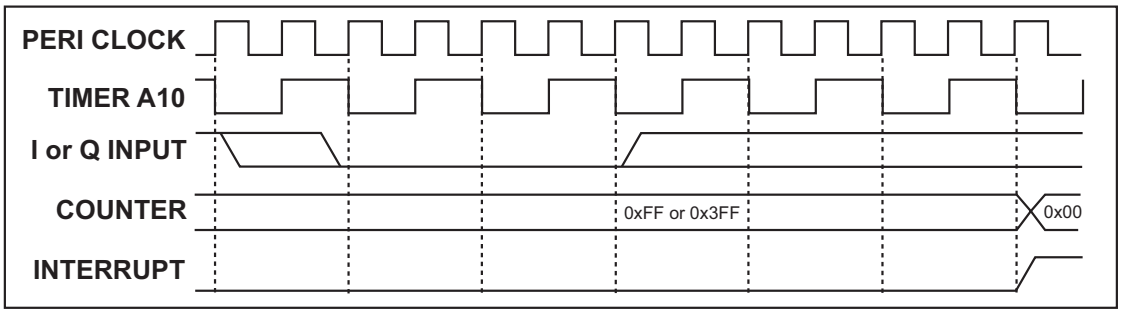
The operation of the counter as a function of the I and Q inputs is shown below.



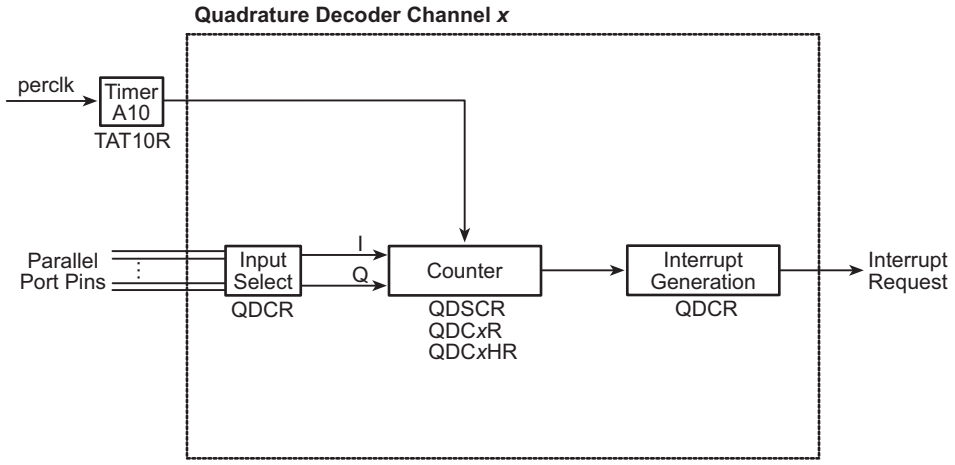
The Quadrature Decoders are clocked by the output of Timer A10, giving a maximum clock rate from perclk/2 down to perclk/512. The time constant of Timer A10 must be fast enough to sample the inputs properly. Both the I and Q inputs go through a digital filter that rejects pulses shorter than two clock period wide. In addition, the clock rate must be high enough that transitions on the I and Q inputs are sampled in different clock cycles. Input capture may be used to measure the pulse width on the I inputs because they come from the odd-numbered port bits. The operation of the digital filter is shown below.



The Quadrature Decoder generates an interrupt when the counter increments from 0xFF (0x3FF in 10-bit mode) to 0x00, or when the counter decrements from 0x00 to 0xFF (0x3FF in 10-bit mode). The timing for the interrupt is shown below. Note that the status bits in the QDCSR are set coincident with the interrupt, and the interrupt and status bits are cleared by reading the QDCSR.



21.1.1 Block Diagram



21.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
Quad Decode Ctrl/Status Register	QDCSR	0x0090	R/W	xxxxxxxx	
Quad Decode Control Register	QDCR	0x0091	W	00xx0000	00000000
Quad Decode Count 1 Register	QDC1R	0x0094	R	xxxxxxxx	
Quad Decode Count 1 High Register*	QDC1HR	0x0095	R	—	xxxxxxxx
Quad Decode Count 2 Register	QDC2R	0x0096	R	xxxxxxxx	
Quad Decode Count 2 High Register*	QDC2HR	0x0097	R	—	xxxxxxxx

* Introduced with Rabbit 3000A chip

21.2 Dependencies

21.2.1 I/O Pins

Each Quadrature Decoder channel can accept the two encoder inputs from one of three different locations, as shown in the table below. Each channel can select a different input option. Note that these pins can be used for other peripherals at the same time as the Quadrature Decoder peripheral; one example of this use is to use measure pulse width on the I channels with the Input Capture peripheral.

Inputs	Channel 1		Channel 2	
	I	Q	I	Q
Option 1	PF1	PF0	PF3	PF2
Option 2	PF5	PF4	PF7	PF6

21.2.2 Clocks

The 8/10-bit Quadrature Decoder counters are clocked from the output of Timer A10, and can run at rates from the peripheral clock divided by 2 down to the peripheral clock divided by 512 by writing the appropriate value to TAT10R. The clock rate must be high enough that transitions on the inputs are sampled in different clock cycles. In addition, both the I and Q inputs go through a digital filter that rejects pulses shorter than two clock periods wide.

21.2.3 Other Registers

Register	Function
TAT10R	Time constant for Quadrature Decoder clock

21.2.4 Interrupts

Each Quadrature Decoder channel can generate an interrupt whenever the counter increments from 0x0FF (0x3FF in 10-bit mode) to 0x00 or when the counter decrements from 0x000 to 0x0FF (0x3FF for 10-bit mode). The interrupt request is cleared when QDCSR is read.

The Quadrature Decoder interrupt vector is in the IIR at offset 0x190. It can be set as Priority 1, 2, or 3.

The status bits in the QDCSR are set coincident with the interrupt request and are reset when QDCSR is read.

21.3 Operation

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

1. Configure Timer A10 via TAT10R to provide the desired Quadrature Decoder clock speed.
2. Configure QDCR to select the input pins for the two channels.
3. Reset the counters by writing to QDCSR.

21.3.1 Handling Interrupts

The following steps explain how an interrupt is set up and used.

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Configure QDCR to select the interrupt priority (note that interrupts will be enabled once this value is set).

The following actions occur within the interrupt service routine.

- Since a Quadrature Decoder interrupt occurs when the counter rolls over, determine exactly why the interrupt occurred by reading the status bits in QDCSR and adjust any software counters accordingly. This will also clear the interrupt request.
- The current counter value can be read from QDCxR (and QDCxHR if the 10-bit counter is enabled).

21.3.2 Example ISR

A sample interrupt handler is shown below.

```
qd_isr::
   push af              ; save used registers
   ioi ld a, (QDCSR)    ; clear the interrupt request and get status

   ; perform any necessary software counter adjustments here
   ; read current counter value(s)

   pop af               ; restore used registers
   ipres
   ret
```

21.4 Register Descriptions

Quad Decode Control/Status Register (QDCSR) (Address = 0x0090)		
Bit(s)	**Value**	**Description**
7 (Read-only)	0	Quadrature Decoder 2 did not increment from 0xFF (0x3FF in 10-bit mode).
	1	Quadrature Decoder 2 incremented from 0xFF (0x3FF in 10-bit mode) to 0x00. This bit is cleared by a read of his register.
6 (Read-only)	0	Quadrature Decoder 2 did not decrement from 0x00.
	1	Quadrature Decoder 2 decremented from 0x00 to 0xFF (0x3FF in 10-bit mode). This bit is cleared by a read of this register.
5		This bit always reads as zero.
4 (Write-only)	0	No effect on Quadrature Decoder 2.
	1	Reset Quadrature Decoder 2 to 0x00 without causing an interrupt.
3 (Read-only)	0	Quadrature Decoder 1 did not increment from 0xFF (0x3FF in 10-bit mode).
	1	Quadrature Decoder 1 incremented from 0xFF (0x3FF in 10-bit mode) to 0x00. This bit is cleared by a read of this register.
2 (Read-only)	0	Quadrature Decoder 1 did not decrement from 0x00.
	1	Quadrature Decoder 1 decremented from 0x00 to 0xFF (0x3FF in 10-bit mode). This bit is cleared by a read of this register.
1		This bit always reads as zero.
0 (Write-only)	0	No effect on Quadrature Decoder 1.
	1	Reset Quadrature Decoder 1 to 0x00 without causing an interrupt.

Quad Decode Control Register (QDCR) (Address = 0x0091)		
Bit(s)	**Value**	**Description**
7:6	00	Disable Quadrature Decoder 2 inputs. Writing a new value to these bits will not cause Quadrature Decoder 2 to increment or decrement.
	01	This bit combination is reserved and should not be used.
	10	Quadrature Decoder 2 inputs from PF2 and PF3.
	11	Quadrature Decoder 2 inputs from PF6 and PF7.
5	0	Eight-bit Quadrature Decoder counters (both channels).
	1*	Ten-bit Quadrature Decoder counters (both channels).
4		This bit is reserved and should be written as zero.
3:2	00	Disable Quadrature Decoder 1 inputs. Writing a new value to these bits will not cause Quadrature Decoder 1 to increment or decrement.
	01	This bit combination is reserved and should not be used.
	10	Quadrature Decoder 1 inputs from PF0 and PF1.
	11	Quadrature Decoder 1 inputs from PF4 and PF5.
1:0	00	Quadrature Decoder interrupts are disabled.
	01	Quadrature Decoder interrupt use Interrupt Priority 1.
	10	Quadrature Decoder interrupt use Interrupt Priority 2.
	11	Quadrature Decoder interrupt use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip

Quad Decode Count Register (QDC1R) (Address = 0x0094) (QDC2R) (Address = 0x0096)		
Bit(s)	**Value**	**Description**
7:0	Read	The current value of bits 7-0 of the Quadrature Decoder counter is reported.

Quad Decode Count High Register (QDC1HR) (Address = 0x0095) (QDC2HR) (Address = 0x0097)		
Bit(s)	**Value**	**Description**
7:2	Read	These bits are reserved and will always read as zeros.
1:0	Read	The current value of bits 9-8 of the Quadrature Decoder counter is reported.

Timer A Time Constant 10 Register (TAT10R) (Address = 0x00AA)		
Bit(s)	**Value**	**Description**
7:0		Time constant for the Timer A counter. This time constant will take effect the next time that the Timer A counter counts down to zero. The timer counts modulo *n* + 1, where *n* is the programmed time constant.

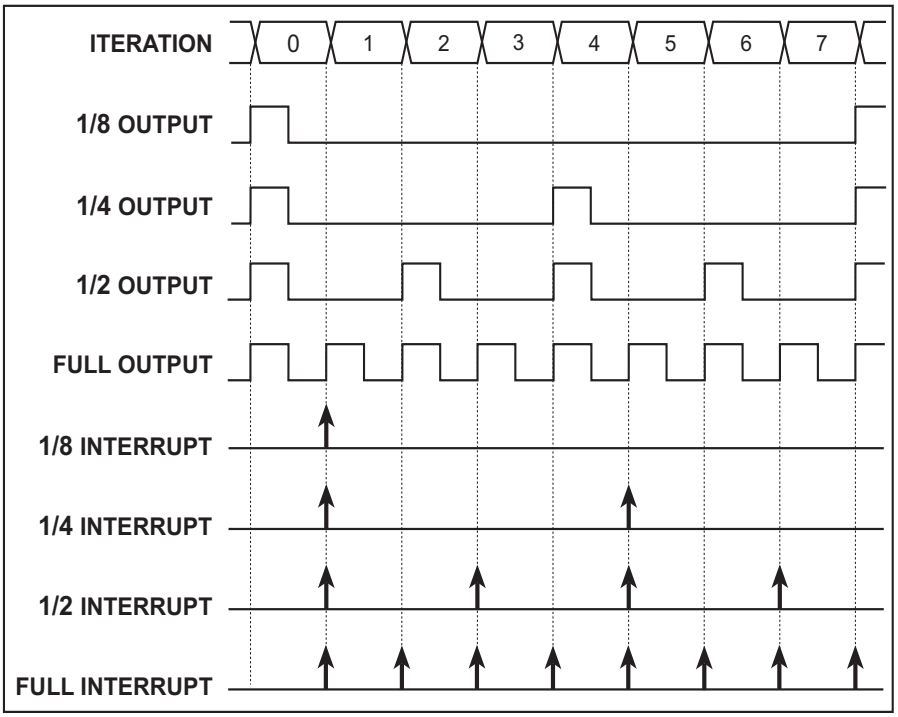
22. PULSE WIDTH MODULATOR

22.1 Overview

The Pulse Width Modulator (PWM) consists of a 10-bit free running counter and four width registers. A PWM output consists of a train of periodic pulses within a 1024-count frame with a duty cycle that varies from 1/1024 to 1024/1024. 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 is clocked by the output of Timer A9 which is used to set the period.

Each PWM output high time can optionally be spread throughout the cycle to reduce ripple on the externally filtered PWM output. The PWM outputs can be passed through a filter and used as a 10-bit D/A converter. The outputs can also be used to directly drive devices such as motors or solenoids that have intrinsic filtering.

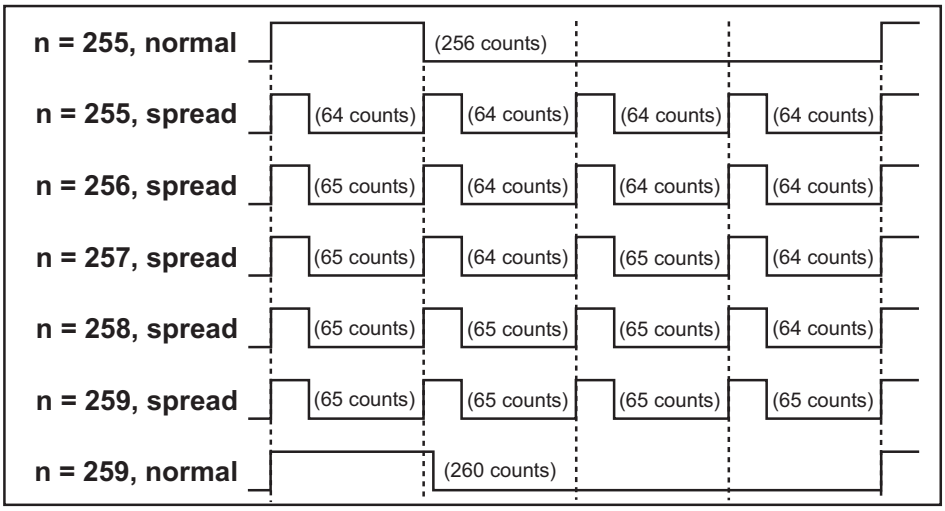
The PWM outputs can trigger a PWM interrupt on every PWM cycle, every other cycle, every fourth cycle, or every eighth cycle. In addition, the PWM output can be suppressed every other cycle, three out of every four cycles, or seven out of every eight cycles. These options provide support for driving servos and to generate audio signals. The setup for this interrupt is done in the PWL0R and PWL1R registers. The timing is shown below.



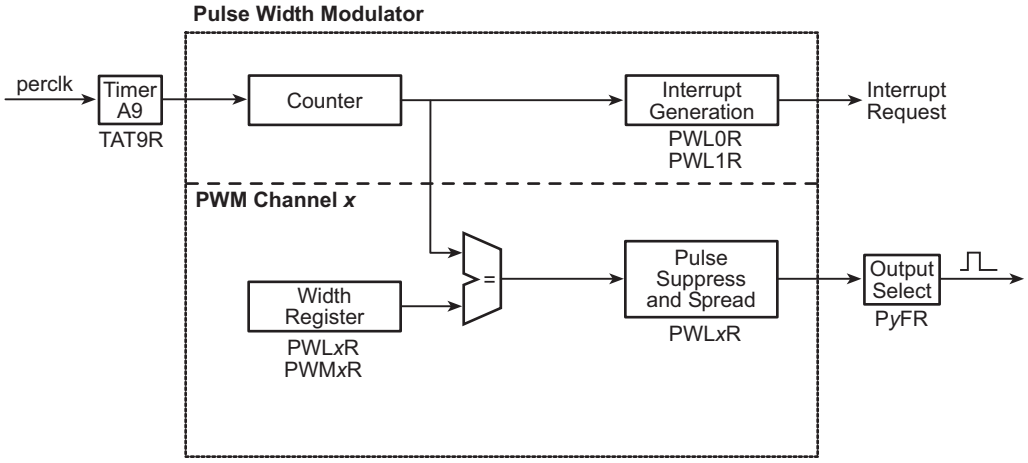
The spreading function is implemented by dividing each 1024-clock cycle into four quadrants of 256 clocks each. Within each quadrant, the Pulse-Width Modulator uses the eight MSBs of each pulse-width register to select the base width in each of the quadrants. This is the equivalent to dividing the contents 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 the table below. The "*n*/4" term is the base count, formed from the eight MSBs of the pulse-width register.

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.



22.1.1 Block Diagram



22.1.2 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
PWM LSB 0 Register	PWL0R	0x0088	W	xxxxxxxx	xxxxx00x
PWM MSB 0 Register	PWM0R	0x0089	W	xxxxxxxx	
PWM LSB 1 Register	PWL1R	0x008A	W	xxxxxxxx	xxxxx00x
PWM MSB 1 Register	PWM1R	0x008B	W	xxxxxxxx	
PWM LSB 2 Register	PWL2R	0x008C	W	xxxxxxxx	xxxxx00x
PWM MSB 2 Register	PWM2R	0x008D	W	xxxxxxxx	
PWM LSB 3 Register	PWL3R	0x008E	W	xxxxxxxx	xxxxx00x
PWM MSB 3 Register	PWM3R	0x008F	W	xxxxxxxx	

22.2 Dependencies

22.2.1 I/O Pins

Each PWM channel can be output on up one of two pins, which can be selected via the parallel port alternate output registers.

PWM	Output Pins
Channel 0	PF4, PG3
Channel 1	PF5, PG7
Channel 2	PF6, PD3
Channel 3	PF7, PD7

22.2.2 Clocks

The PWM counter is clocked from the output of Timer A9, and can run at rates from perclk/2 down to perclk/512 by writing the appropriate value to TAT9R.

22.2.3 Other Registers

Register	Function
TAT9R	Time constant for PWM clock
PDFR, PFFR, PGFR	Alternate port output selection

22.2.4 Interrupts

The PWM can generate an interrupt for every PWM counter rollover, every second rollover, every fourth rollover, or every eighth rollover. This option is selected in PWL1R. The interrupt request is cleared by a write to any PWM register.

The PWM interrupt vector is in the IIR at offset 0x170. It can be set as Priority 1, 2, or 3 by writing to PWL0R.

22.3 Operation

The following steps explain how to set up a PWM channel.

1. Configure Timer A9 via TAT9R to provide the desired PWM clock frequency.
2. Configure PWLxR to select whether to spread the PWM output throughout the cycle.
3. Configure PWLxR to select whether to suppress the PWM output.
4. Configure the duty cycle by writing to PWLxR and PWMxR.

22.3.1 Handling Interrupts

The following steps explain how an interrupt is set up and used.

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Configure PWL0R to select the PWM interrupt priority and PWL1R to select PWM interrupt suppression (if an interrupt is desired).

The following actions occur within the interrupt service routine.

- Any PWM values may be updated.
- The interrupt request should be cleared by writing to any PWM register.

22.3.2 Example ISR

A sample interrupt handler is shown below.

```
pwm_isr::
   push af              ; save used registers
   ld a, 0x55
   ioi ld (PWM0R), a     ; update a PWM value
   ; note that interrupt request is also cleared by register write above
   pop af               ; restore used registers
   ipres
   ret
```

22.4 Register Descriptions

PWM LSB x Register		(PWL0R) (Address = 0x0088) (PWL1R) (Address = 0x008A) (PWL2R) (Address = 0x008C) (PWL3R) (Address = 0x008E)
Bit(s)	**Value**	**Description**
7:6	Write	The least significant two bits for the Pulse Width Modulator count are stored.
5:4*	00	Normal PWM operation.
	01	Suppress PWM output seven out of eight iterations of PWM counter.
	10	Suppress PWM output three out of four iterations of PWM counter.
	11	Suppress PWM output one out of two iterations of PWM counter.
3*		This bit is ignored and should be written with zero.
2:1*	00	Pulse Width Modulator interrupts are disabled.
	01	Pulse Width Modulator interrupts use Interrupt Priority 1.
	10	Pulse Width Modulator interrupts use Interrupt Priority 2.
	11	Pulse Width Modulator interrupts use Interrupt Priority 3.
0	0	PWM output High for single block.
	1	Spread PWM output throughout the cycle.

* Bits 5:1 are ignored in the original Rabbit 3000 microprocessor.

PWM MSB x Register		(PWM0R) (Address = 0x0089) (PWM1R) (Address = 0x008B) (PWM2R) (Address = 0x008D) (PWM3R) (Address = 0x008F)
Bit(s)	**Value**	**Description**
7:0	Write	Most significant eight bits for the Pulse Width Modulator count. With a count of "*n*", the PWM output will be High for "*n* + 1" clocks out of the 1024 clocks of the PWM counter.

Timer A Time Constant 9 Register	(TAT9R)	(Address = 0x00A8)
Bit(s)	**Value**	**Description**
7:0		Time constant for the Timer A counter. This time constant will take effect the next time that the Timer A counter counts down to zero. The timer counts modulo *n* + 1, where *n* is the programmed time constant.

23. EXTERNAL I/O CONTROL

23.1 Overview

The Rabbit 3000's external I/O space consists of 64 KB that is accessed by prefixing a read or write instruction with the IOE instruction. These accesses can go onto the memory bus or onto the external I/O bus (described below). There are three dedicated signal pins (/IORD, /IOWR, /BUFEN) that toggle for all external I/O accesses, and eight I/O strobes that can be associated with this external I/O space and directed out of Parallel Port E.

23.1.1 External I/O Bus

The Rabbit 3000 can enable a separate external I/O bus for external devices to keep bus loading on the memory bus at an acceptable level. This bus consists of eight data lines on Parallel Port A and up to six address lines on Parallel Port B. This functionality is mutually exclusive with the slave port and regular parallel I/O on Parallel Ports A and B.

When enabled, the address lines of the external I/O bus hold their value until a new value is written to them. The data lines return to a tristate mode after each transaction.

See Section 23.1.2 for memory timing for external I/O accesses.

23.1.2 I/O Strobes

There are eight I/O strobes available in the Rabbit 3000. Each has a separate 8 KB address range that can be enabled as a chip select, read strobe, write strobe, or a read/write strobe. The number of wait states can be set to 1, 3, 7, or 15, and the signal can be active high or low.

Table 23-1. External I/O Strobes

Register	Port E Pin	I/O Address A[15:13]	External I/O Address Range
IB0CR	PE0	000	0x0000–0x1FFF
IB1CR	PE1	001	0x2000–0x3FFF
IB2CR	PE2	010	0x4000–0x5FFF
IB3CR	PE3	011	0x6000–0x7FFF
IB4CR	PE4	100	0x8000–0x9FFF
IB5CR	PE5	101	0xA000–0xBFFF
IB6CR	PE6	110	0xC000–0xDFFF
IB7CR	PE7	111	0xE000–0xFFFF

The I/O strobes can be used for devices on the memory bus or the external I/O bus, and can be enabled to go out on the memory bus alone or both buses. It is also possible to shorten the read strobe by one clock cycle by pulling in the trailing edge, which guarantees one clock cycle of hold time for transactions.

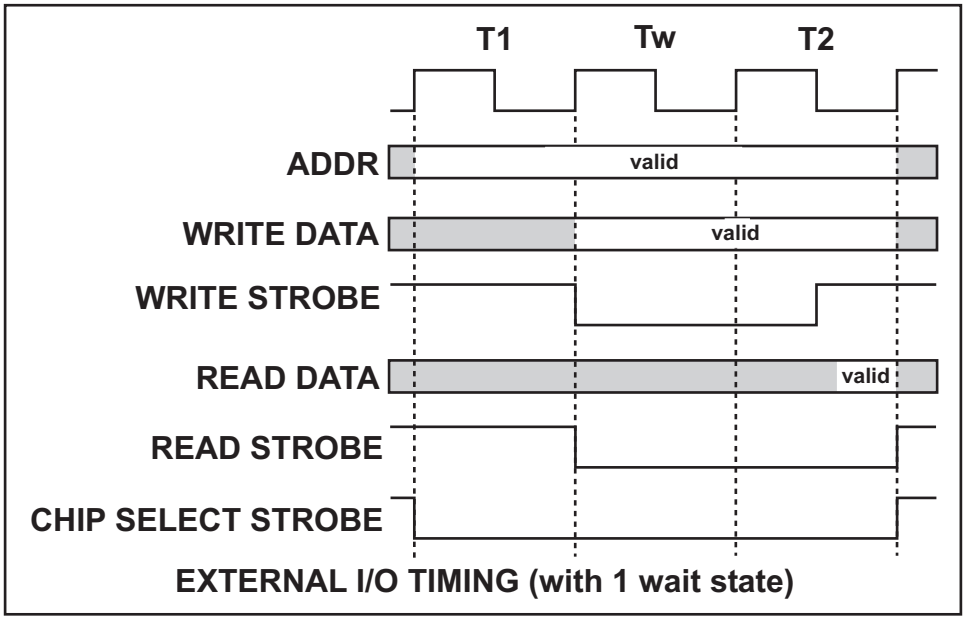
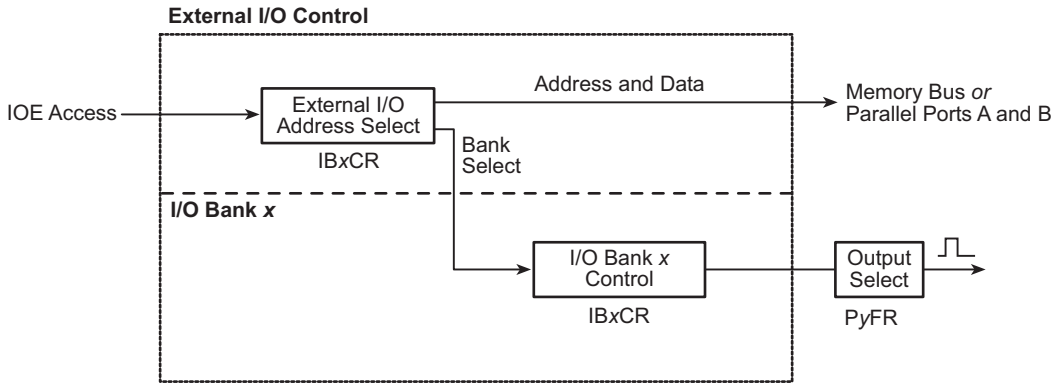


Figure 23-1. External I/O Bus Cycles

The strobes can be enabled to come out on Parallel Port E.

By default the I/O strobes are configured as read-only chip selects with 15 wait states and normal timing. These settings will affect the /IORD, /IOWR, and /BUFEN signals for external I/O writes even if no other strobe outputs are enabled in the parallel port registers.

23.1.3 Block Diagram



23.1.4 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
I/O Bank 0 Control Register	IB0CR	0x0080	W	000000xx	00000000
I/O Bank 1 Control Register	IB1CR	0x0081	W	000000xx	00000000
I/O Bank 2 Control Register	IB2CR	0x0082	W	000000xx	00000000
I/O Bank 3 Control Register	IB3CR	0x0083	W	000000xx	00000000
I/O Bank 4 Control Register	IB4CR	0x0084	W	000000xx	00000000
I/O Bank 5 Control Register	IB5CR	0x0085	W	000000xx	00000000
I/O Bank 6 Control Register	IB6CR	0x0086	W	000000xx	00000000
I/O Bank 7 Control Register	IB7CR	0x0087	W	000000xx	00000000

23.2 Dependencies

23.2.1 I/O Pins

The external I/O bus uses PA0–PA7 for data, and PB2–PB7 for address lines, depending on the setting in SPCR.

The /IOWR, /IORD, and /BUFEN pins are dedicated strobes for external I/O accesses.

The I/O strobes can be directed out to pins on Parallel Port E; each bank can be directed to the appropriate pin (bank zero on PE0; bank one on PE1; etc.). The strobes will affect outputs on /IOWR, /IORD, and /BUFEN at all times.

23.2.2 Clocks

All external I/O accesses and strobes are based on the processor clock.

23.2.3 Other Registers

Register	Function
SPCR	Enable the external I/O bus.
PCFR, PCALR, PCAHR PDFR, PDALR, PDAHR, PEFR, PEALR, PEAHR	Select Parallel Port C, D, or E pins as I/O strobe outputs. Select PD1, PD3, PD5, or PD7 as address bits 6-7.

23.2.4 Interrupts

There are no interrupts associated with external I/O.

23.3 Operation

23.3.1 External I/O Bus

The following steps must be taken before using external I/O bus:

1. Enable the external I/O bus by writing to SPCR.
2. Set the I/O timing for a particular device by writing to the appropriate IB*x*CR register for the I/O bank desired.

Once the external I/O bus is enabled, all memory read/write instructions prefixed with an IOE will go to either the memory bus or external I/O bus, depending on the setup in that bank's IB*x*CR register.

23.3.2 I/O Strobes

The following step must be taken before using an I/O strobe:

1. Set the strobe type and timing for a particular device by writing to the appropriate IB*x*CR register for the I/O bank desired.

On startup, the I/O strobes are set as chip selects with 15 wait states, read-only, active-low signaling, and will use the external I/O bus. These settings will be used for the dedicated I/O strobe pins /IORD, /IOWR, and /BUFEN whenever an external I/O write occurs even if not I/O strobe signals are being output on parallel port pins.

23.4 Register Descriptions

I/O Bank x Control Register		(IB0CR) (Address = 0x0080) (IB1CR) (Address = 0x0081) (IB2CR) (Address = 0x0082) (IB3CR) (Address = 0x0083) (IB4CR) (Address = 0x0084) (IB5CR) (Address = 0x0085) (IB6CR) (Address = 0x0086) (IB7CR) (Address = 0x0087)
Bit(s)	**Value**	**Description**
7:6	00	Fifteen wait states for accesses in this bank.
	01	Seven wait states for accesses in this bank.
	10	Three wait states for accesses in this bank.
	11	One wait state for accesses in this bank.
5:4	00	The I signal is an I/O chip select.
	01	The I signal is an I/O read strobe.
	10	The I signal is an I/O write strobe.
	11	The I signal is an I/O data (read or write) strobe.
3	0	Writes are not allowed to this bank. Transactions are normal in every other way; only the write strobe is inhibited.
	1	Writes are allowed to this bank.
2	0	I/O strobe (I) is active low.
	1	I/O strobe (I) is active high.
1*	0	Normal I/O Transaction timing.
	1	Shorten read strobe by one clock cycle. Transaction length remains the same.
0*	0	Use I/O bus if enabled.
	1	Always use memory data bus.

* These bits are ignored in versions of the Rabbit 3000 before the Rabbit 3000A and were always written with zero.

Slave Port Control Register (SPCR) (Address = 0x0024)		
Bit(s)	**Value**	**Description**
7	0	Program fetch as a function of the SMODE pins.
	1	Ignore the SMODE pins program fetch function.
6:5	Read	These bits report the state of the SMODE pins.
	Write	These bits are ignored and should be written with zero.
4	0	/SCS from PE7.
	1	/SCS from PB6*.
3:2 (Write only)	00	Disable the slave port. Parallel Port A is a byte-wide input port.
	01	Disable the slave port. Parallel Port A is a byte-wide output port.
	10	Enable the slave port.
	11	Enable the external I/O bus. Parallel Port A is used for the data bus and Parallel Port B[7:2] is used for the address bus.
1:0 (Write only)	00	Slave port interrupts are disabled.
	01	Slave port interrupts use Interrupt Priority 1.
	10	Slave port interrupts use Interrupt Priority 2.
	11	Slave port interrupts use Interrupt Priority 3.

* Introduced with Rabbit 3000A chip.

24. BREAKPOINTS

24.1 Overview

The Breakpoint/Debug controller allows the RST 28 instruction to be used as a software breakpoint. Normally the RST 28 instruction causes a call to a particular location in memory, but the operation of this instruction is modified when the breakpoint/debug feature is enabled. The RST 28 instruction is treated as a NOP in the breakpoint/debug mode.

Another breakpoint feature is the ability to disable the RST 28 instruction. The RST 28 vector was often used as a breakpoint feature by adding that instruction to code; by enabling a bit in BDCR, the RST 28 instruction will execute as a NOP instead, providing an easy way to disable that type of breakpoint.

24.1.1 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
Breakpoint/Debug Control Register	BDCR	0x001C	W	0xxxxxxx	00000000

24.2 Dependencies

24.2.1 I/O Pins

There are no I/O pins associated with breakpoints.

24.2.2 Clocks

There are no clocks associated with breakpoints.

24.2.3 Other Registers

There are no other registers associated with breakpoints.

24.2.4 Interrupts

There are no interrupts associated with breakpoints.

24.3 Register Descriptions

Breakpoint/Debug Control Register (BDCR) (Address = 0x001C)		
Bit(s)	**Value**	**Description**
7	0	Normal RST 28 operation.
	1	RST 28 is NOP.
6:0		These bits are reserved and should be written with zeros.

25. LOW-POWER OPERATION

25.1 Overview

The Rabbit 3000 contains several power-saving features. Since the power consumed by the processor is proportional to the clock speed, the Rabbit 3000 provides 12 clock modes that can go as low as 2 kHz. To further reduce power consumption in those ultra-sleepy modes, various shortened chip select strobes are available to reduce current draw by the attached memory devices.

Figure 25-1 shows a typical current draw as a function of the main clock frequency. The values shown do not include any current consumed by external oscillators or memory. It is assumed that approximately 30 pF is connected to each address line.

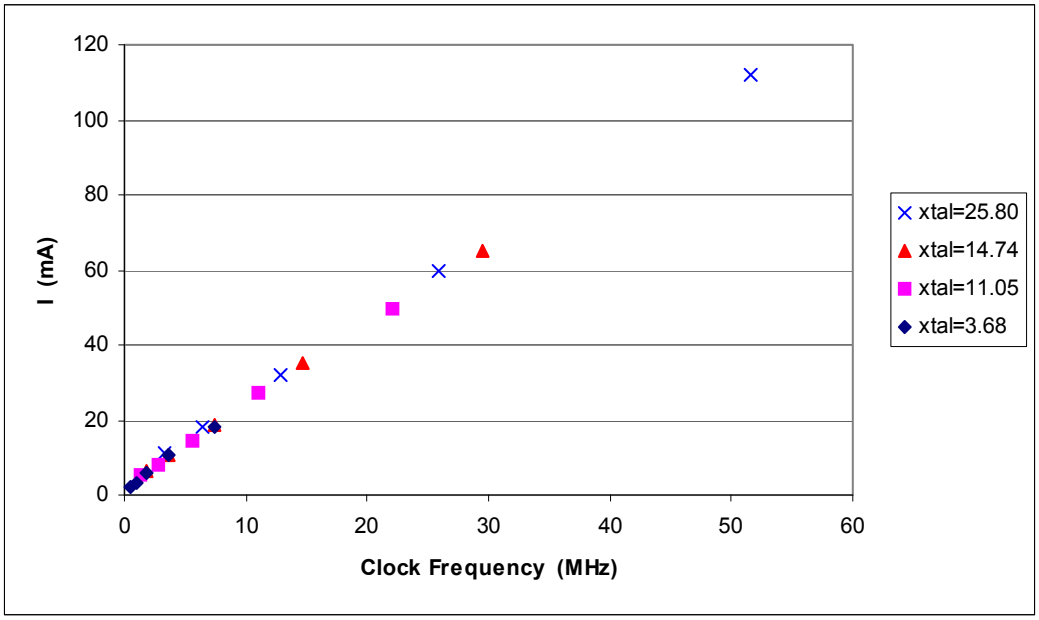


Figure 25-1. Rabbit 3000 System Current vs. Frequency at 3.3 V

Figure 25-2 shows a typical current draw for the sleepy modes.

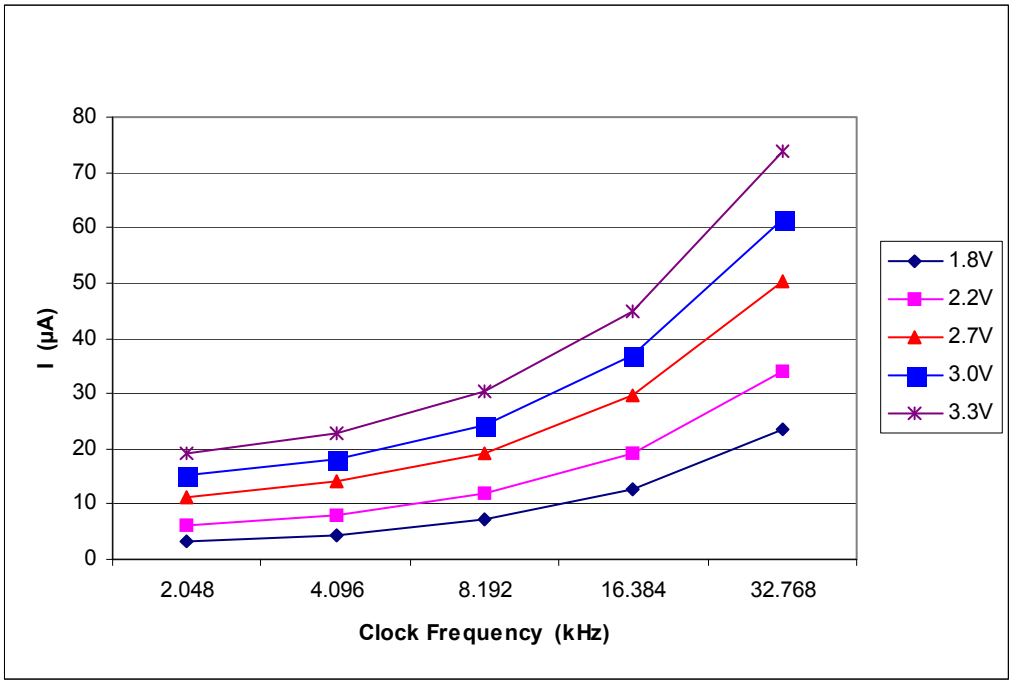


Figure 25-2. Sleepy Mode Current Consumption

25.1.1 Registers

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
Global Control/Status Register	GCSR	0x0000	R/W	11000000	
Global Power Save Control Register	GPSCR	0x000D	W	0000x000	00000000
Global Clock Double Register	GCDR	0x000F	W	00000000	

25.2 Operation

25.2.1 Unused Pins

Input (or bidirectional) pins that are unused in a design can pick up noise that may cause the transistors in the input buffer to switch states quickly, causing unnecessary current draw. To avoid this, all unused pins should be connected to a weak pullup or pulldown resistor (approximately 100 kΩ) and left as inputs. This provides protection from noise when the pin is an input, but also limits the current draw if the pin gets inadvertently enabled as an output.

25.2.2 Clock Rates

The processor and peripheral clocks in the Rabbit 3000 can be run in six different modes using the main oscillator: full speed; divided by 2, 4, 6, or 8; and the processor clock divided by 8 with the peripheral clock at full speed. If the clock doubler is enabled, the options also include twice the main oscillator frequency and the main oscillator divided by 3.

In addition, the 32 kHz clock can be used for the processor and peripheral clocks; the 32 kHz clock can also be divided by 2, 4, 8, or 16, which provides dramatically lower power consumption.

Table 25-1 lists the options for the clock modes and the processor clock frequency.

Table 25-1. Clock Modes

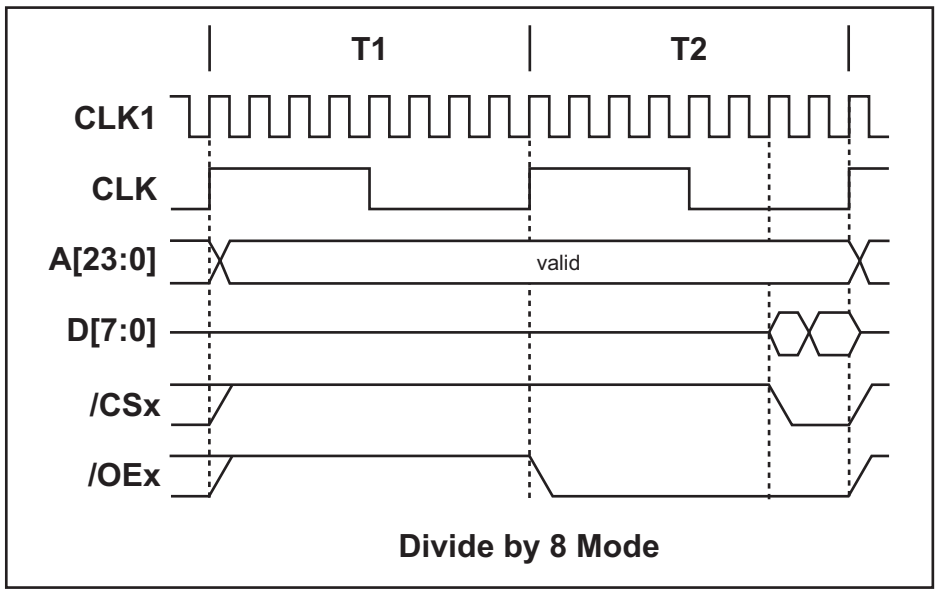
<table>
<tr><th>Main Oscillator GCSR Setting</th><th>Clock Doubler</th><th>32 kHz Divider</th><th>Processor Clock Frequency</th></tr>
<tr><td>Full</td><td>On</td><td rowspan="10">N/A</td><td>2 × Main Oscillator</td></tr>
<tr><td>Full</td><td>Off</td><td rowspan="2">Main Oscillator</td></tr>
<tr><td>Divided by 2</td><td>On</td></tr>
<tr><td>Divided by 2</td><td>Off</td><td rowspan="2">Main Oscillator / 2</td></tr>
<tr><td>Divided by 4</td><td>On</td></tr>
<tr><td>Divided by 6</td><td>On</td><td>Main Oscillator / 3</td></tr>
<tr><td>Divided by 4</td><td>Off</td><td rowspan="2">Main Oscillator / 4</td></tr>
<tr><td>Divided by 8</td><td>On</td></tr>
<tr><td>Divided by 6</td><td>Off</td><td>Main Oscillator / 6</td></tr>
<tr><td>Divided by 8</td><td>Off</td><td>Main Oscillator / 8</td></tr>
<tr><td rowspan="5">Off
(32 kHz divider used)</td><td rowspan="5">N/A</td><td>Disabled</td><td>32.768 kHz</td></tr>
<tr><td>/ 2</td><td>16.384 kHz</td></tr>
<tr><td>/ 4</td><td>8.192 kHz</td></tr>
<tr><td>/ 8</td><td>4.096 kHz</td></tr>
<tr><td>/ 16</td><td>2.048 kHz</td></tr>
</table>

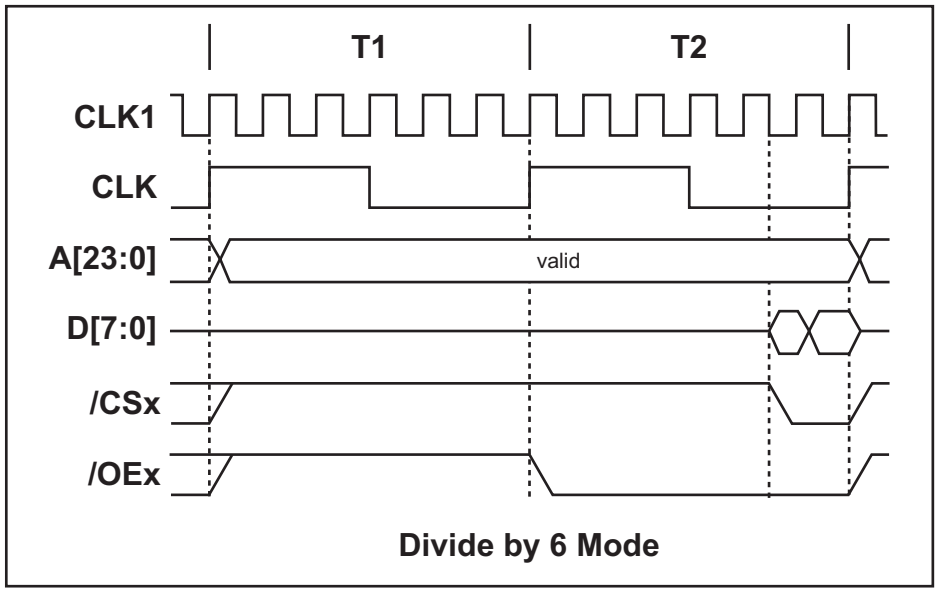
Depending on the application, the processor can continue executing code normally when the main oscillator is divided down to a lower value. However, when the processor clock is running off the 32 kHz clock, it is recommended that the Rabbit 3000 be performing a tight polling loop, waiting for a wakeup event.

25.2.3 Short Chip Selects

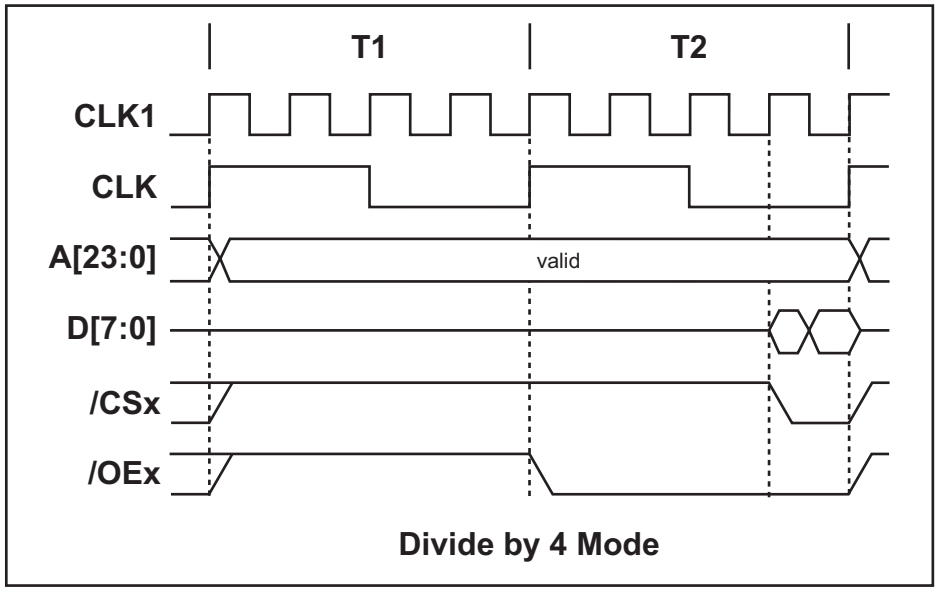
When running at a reduced clock speed, it is likely that the chip selects for external devices will not need to be active for an entire clock cycle. By reducing the width of the chip select, the power consumption of the memory chip can be reduced without having any affect on the processor itself.

For reduced processor speeds based on the main oscillator, a short chip select can be enabled in GPSCR (this feature is not available when the processor is running at full speed). This feature can be enabled separately for both reads and writes. When enabled, the chip select signals will be the width of two undivided clocks and located at the end of the transaction. The read data in the figures below is sampled by the rising edge of CLKI that terminated the T2 cycle. Wait states are inserted between T1 and T2 so they do not affect the width of the strobe.

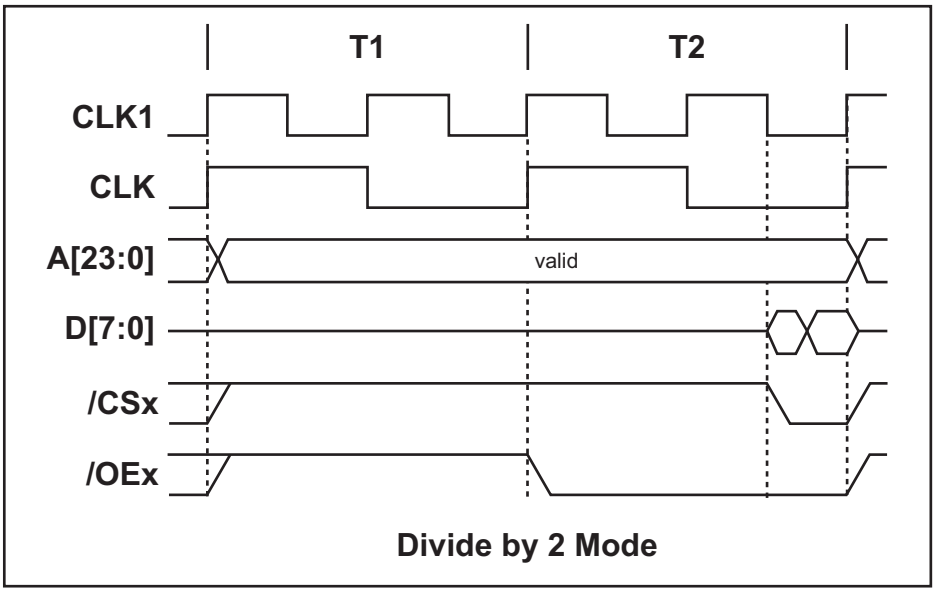




Divide by 6 Mode

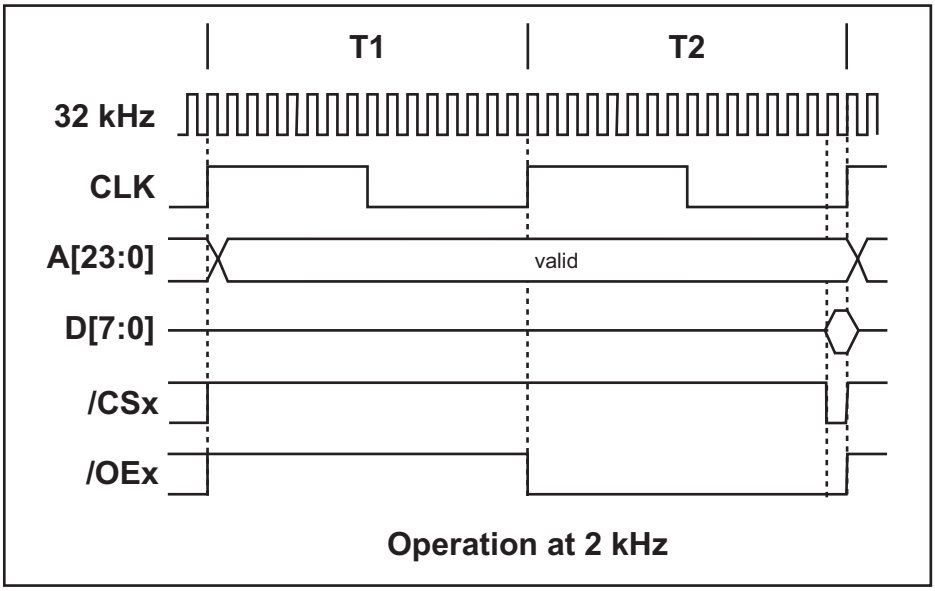


Divide by 4 Mode

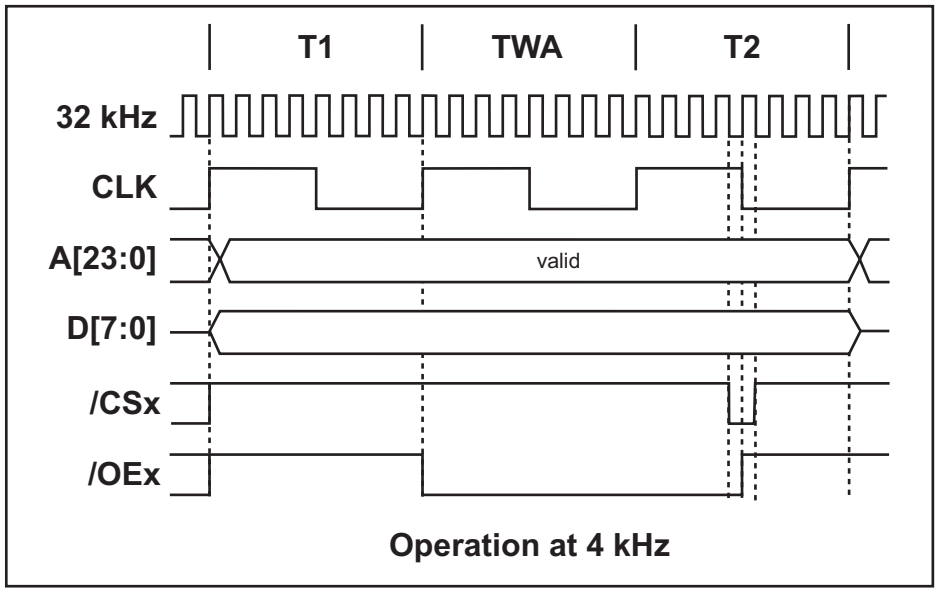


Divide by 2 Mode

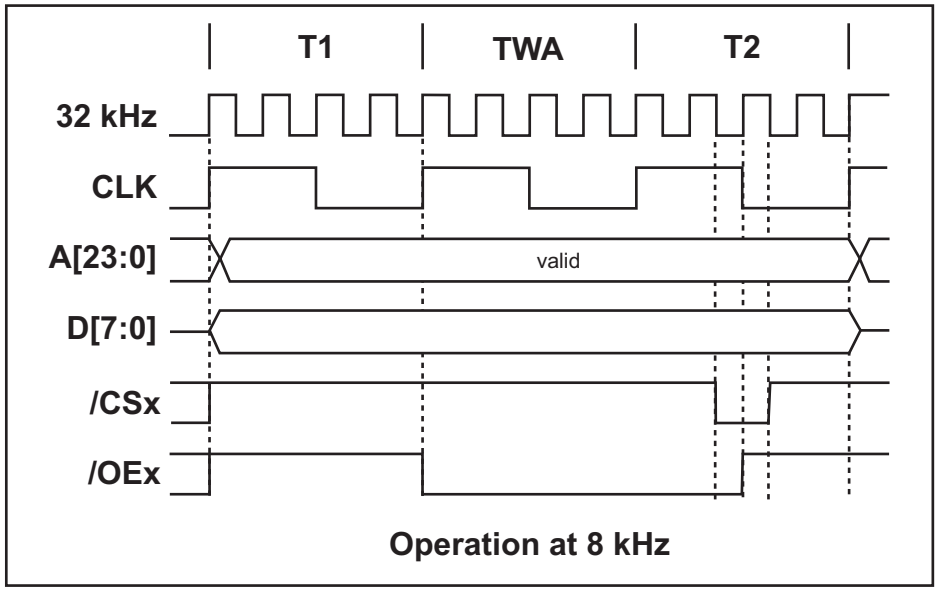
When the processor is running off the 32 kHz clock, the short chip select option will produce chip select signal that is the width of a single 32 kHz clock (30.5 microseconds); otherwise the timing is identical to the short chip select options based off the main oscillator. Read strobe figures are shown below.



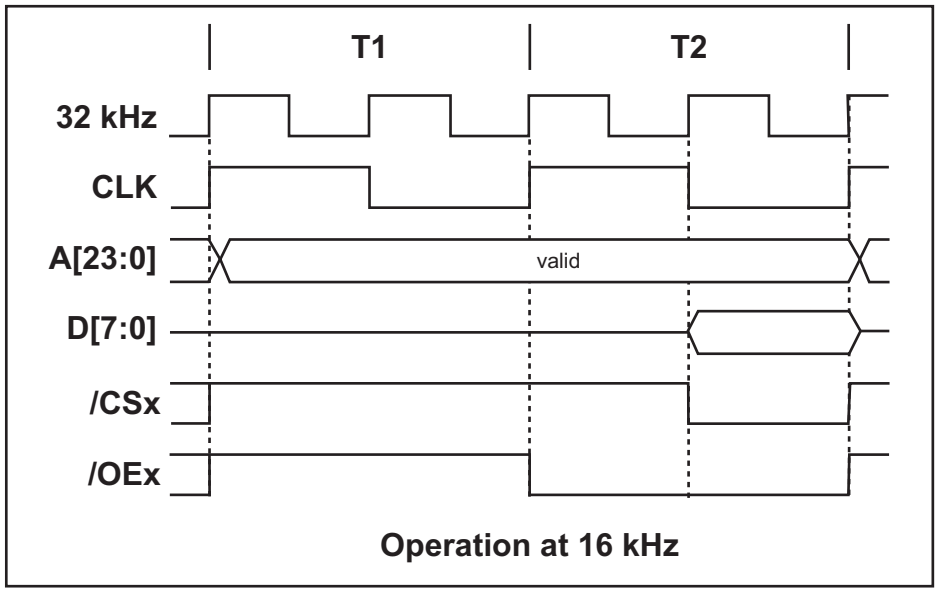
Operation at 2 kHz



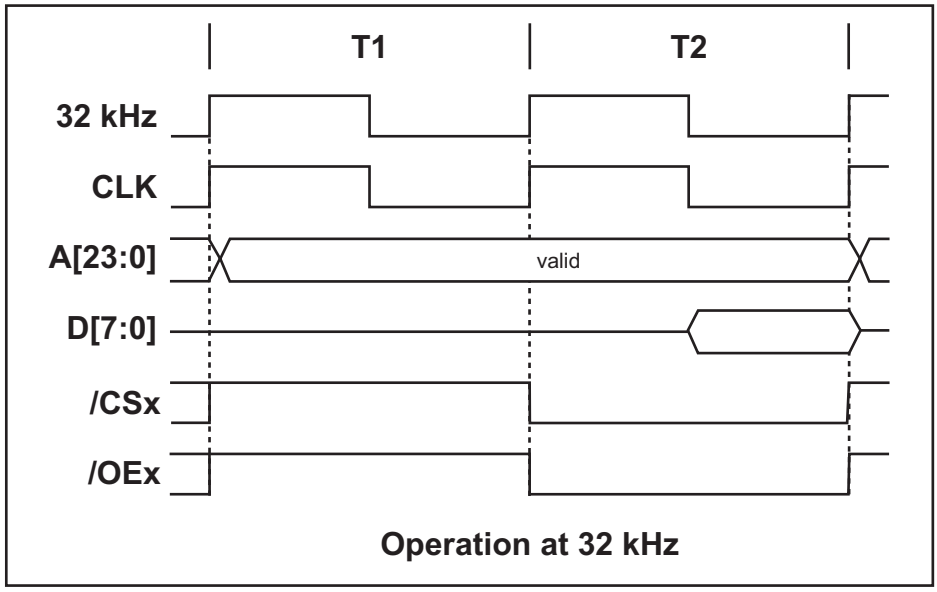
Operation at 4 kHz



Operation at 8 kHz



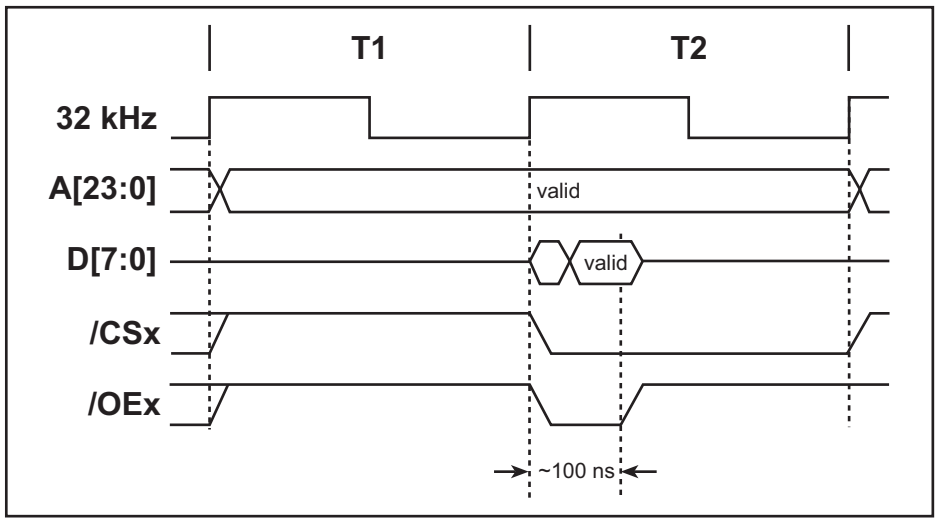
Operation at 16 kHz



Operation at 32 kHz

25.2.4 Self-Timed Chip Selects

Self-timed chip selects can be enabled via GPSCR to reduce power consumption even more when running off the 32kHz oscillator. When self-timed chip selects are enabled, the chip select is only active for a short (selectable) period of time. A sample read and write timing diagram is shown below.



25.3 Register Descriptions

Global Control/Status Register (GCSR) (Address = 0x0000)		
Bit(s)	**Value**	**Description**
7:6 (Read-only)	00	No reset or watchdog timer timeout since the last read.
	01	The watchdog timer timed out. These bits are cleared by a read of this register.
	10	This bit combination is not possible.
	11	Reset occurred. These bits are cleared by a read of this register.
5	0	No effect on the periodic interrupt. This bit will always be read as zero.
	1	Force a periodic interrupt to be pending.
4:2	xxx	See table below to decode this field.
1:0	00	Periodic interrupts are disabled.
	01	Periodic interrupts use Interrupt Priority 1.
	10	Periodic interrupts use Interrupt Priority 2.
	11	Periodic interrupts use Interrupt Priority 3.

GCSR Clock Select Field

Clock Select Bits 4:2 GCSR	CPU Clock	Peripheral Clock	Main Oscillator	Power-Save CS if Enabled by GPSCR
000	osc/8	osc/8	on	short CS option
001	osc/8	osc	on	short CS option
010	osc	osc	on	none
011	osc/2	osc/2	on	short CS option*
100	32 kHz or fraction	32 kHz or fraction	on	self-timed option short CS option*
101	32 kHz or fraction	32KHz or fraction	off	self-timed option short CS option*
110	osc/4	osc/4	on	short CS option
111	osc/6	osc/6	on	short CS option

* Introduced with Rabbit 3000A chip

Global Power Save Control Register (GPSCR) (Address = 0x000D)		
Bit(s)	**Value**	**Description**
7:5	000	Self-timed chip selects are disabled.
	001	This bit combination is reserved and should not be used.
	010	This bit combination is reserved and should not be used.
	011	This bit combination is reserved and should not be used..
	100	296 ns self-timed chip selects (192 ns best case, 457 ns worst case).
	101	234 ns self-timed chip selects (151 ns best case, 360 ns worst case).
	110	171 ns self-timed chip selects (111 ns best case, 264 ns worst case).
	111	109 ns self-timed chip selects (71 ns best case, 168 ns worst case).
4	0	Normal Chip Select timing for read cycles.
	1	Short Chip Select timing for read cycles (not available in full speed).
3*	0	Normal Chip Select timing for write cycles
	1	Short Chip Select timing for write cycles (not available in full speed).
2:0	000	The 32 kHz clock divider is disabled.
	001	This bit combination is reserved and should not be used.
	010	This bit combination is reserved and should not be used.
	011	This bit combination is reserved and should not be used.
	100	32 kHz clock divided by 2 (16.384 kHz).
	101	32 kHz clock divided by 4 (8.192 kHz).
	110	32 kHz clock divided by 8 (4.096 kHz).
	111	32 kHz clock divided by 16 (2.048 kHz).

* Bit 3 is not used on the original Rabbit 3000 microprocessor.

Global Clock Double Register (GCDR) (Address = 0x000F)		
Bit(s)	**Value**	**Description**
7:4		These bits are reserved and should be written with zeros.
3:0	0000	The clock doubler circuit is disabled.
	0001	6 ns nominal low time.
	0010	7 ns nominal low time.
	0011	8 ns nominal low time.
	0100	9 ns nominal low time.
	0101	10 ns nominal low time.
	0110	11 ns nominal low time.
	0111	12 ns nominal low time.
	1000	13 ns nominal low time.
	1001	14 ns nominal low time.
	1010	15 ns nominal low time.
	1011	16 ns nominal low time.
	1100	17 ns nominal low time.
	1101	18 ns nominal low time.
	1110	19 ns nominal low time.
	1111	20 ns nominal low time.

26. SYSTEM/USER MODE

26.1 Overview

The Rabbit 3000A is the first Rabbit microprocessor to incorporate support for two tiers of control in the processor: *System Mode*, which provides full access to all processor resources; and *User Mode*, a more restricted mode. Table 26-1 describes the essential differences between the System Mode and the User Mode. The System Mode is essentially the same as the normal operation when the System/User Mode is disabled.

Table 26-1. Differences Between System Mode and User Mode

System Mode	User Mode
All peripherals accessible.	No peripherals accessible by default.
All processor control registers available.	No processor control registers available.
All interrupt priorities available.	Interrupt Priority 3 not allowed.
IDET instruction has no effect.	IDET instruction causes Priority 3 “System mode violation” interrupt.
No write protection when 0x00 is written to WPCR (write protection in User mode only)	Write to protected segment causes Priority 3 “write protection violation” interrupt.
Easy to enter User mode (SETUSR instruction).	Difficult to enter system mode (requires interrupt, SYSCALL, or RST instruction).

The main intent of the System/User Mode is to protect critical code (for example, code that performs remote firmware updates), data, and the current processor state (memory setup, peripheral control, etc.) from inadvertent changes by the user’s standard code. By removing access to the processor’s I/O registers and preventing memory writes to critical regions, the user’s code can run without the danger of locking up the processor to the point where it cannot be restarted remotely and/or new code uploaded.

26.1.1 Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Enable Dual-Mode Register	EDMR	0x0420	W	00000000
Real Time Clock User Enable Register	RTUER	0x0300	W	00000000
Slave Port User Enable Register	SPUER	0x0320	W	00000000
Parallel Port A User Enable Register	PAUER	0x0330	W	00000000
Parallel Port B User Enable Register	PBUER	0x0340	W	00000000
Parallel Port C User Enable Register	PCUER	0x0350	W	00000000
Parallel Port D User Enable Register	PDUER	0x0360	W	00000000
Parallel Port E User Enable Register	PEUER	0x0370	W	00000000
Parallel Port F User Enable Register	PFUER	0x0338	W	00000000
Parallel Port G User Enable Register	PGUER	0x0348	W	00000000
Input Capture User Enable Register	ICUER	0x0358	W	00000000
I/O Bank User Enable Register	IBUER	0x0380	W	00000000
PWM User Enable Register	PWUER	0x0388	W	00000000
Quad Decode User Enable Register	QDUER	0x0390	W	00000000
External Interrupt User Enable Register	IUER	0x0398	W	00000000
Timer A User Enable Register	TAUER	0x03A0	W	00000000
Timer B User Enable Register	TBUER	0x03B0	W	00000000
Serial Port A User Enable Register	SAUER	0x03C0	W	00000000
Serial Port B User Enable Register	SBUER	0x03D0	W	00000000
Serial Port C User Enable Register	SCUER	0x03E0	W	00000000
Serial Port D User Enable Register	SDUER	0x03F0	W	00000000
Serial Port E User Enable Register	SEUER	0x03C8	W	00000000
Serial Port F User Enable Register	SFUER	0x03D8	W	00000000

26.2 Dependencies

26.2.1 I/O Pins

There are no pin dependencies for the System/User Mode.

26.2.2 Clocks

There are no clock dependencies for the System/User Mode.

26.2.3 Other Registers

Any writes to the internal I/O registers listed in Table 26-2 are ignored when the System/User Mode is enabled and the processor is in the User Mode.

Table 26-2. I/O Addresses Inaccessible in User Mode

Register Name	Mnemonic	I/O Address
Global Control/Status Register	GCSR	0x0000
Watchdog Timer Control Register	WDTCR	0x0008
Watchdog Timer Test Register	WDTTR	0x0009
Global Clock Modulator 0 Register	GCM0R	0x000A
Global Clock Modulator 1 Register	GCM1R	0x000B
Secondary Watchdog Timer Register*	SWDTR	0x000C
Global Power Save Control Register	GPSCR	0x000D
Global Output Control Register	GOCR	0x000E
Global Clock Double Register	GCDR	0x000F
MMU Instruction/Data Register	MMIDR	0x0010
Stack Segment Register	STACKSEG	0x0011
Data Segment Register	DATASEG	0x0012
Segment Size Register	SEGSIZE	0x0013
Memory Bank 0 Control Register	MB0CR	0x0014
Memory Bank 1 Control Register	MB1CR	0x0015
Memory Bank 2 Control Register	MB2CR	0x0016
Memory Bank 3 Control Register	MB3CR	0x0017
MMU Expanded Code Register	MECR	0x0018
Memory Timing Control Register	MTCR	0x0019
Breakpoint/Debug Control Register	BDCR	0x001C
User Enable registers*		0x03xx
Memory Protection registers*		0x04xx

* These registers are only available on the Rabbit 3000A.

26.2.4 Interrupts

The System Mode Violation interrupt occurs whenever the **IDET** instruction is executed while the System/User mode is enabled and the processor is in the User Mode. Its purpose is to trap when system code is being executed while the processor is in the User Mode.

The System Mode Violation interrupt vector is in the IIR at offset 0x180. It always occurs at Priority 3.

Note that Priority 3 is not available while the System/User Mode is enabled and the processor is in the User Mode. If the processor is placed into Priority 3 either by an instruction or an interrupt, it will respond as if it was set to Priority 2.

When the System/User Mode is enabled, it is critical to handle the SU stack in interrupts as well as the IP stack; always perform a **SURES** before the **IPRES** at the end of the interrupt.

26.3 Operation

The System/User Mode is designed to work with the memory and stack protection features of the Rabbit 3000 processor to provide a seamless framework for protection of critical code. However, there are many levels at which the System/User Mode can be used — some examples are described here.

26.3.1 Memory Protection Only

At the beginning of the user program, all necessary peripherals are enabled, all peripheral interrupts to be used are set up for the User Mode, critical memory regions are protected, stack limits are set, and the various system/memory/stack violation interrupts are enabled. The processor then enters the User Mode and remains in the User Mode for all operations (interrupts can be handled however the user desires). Obviously the critical interrupts can be handled in the System Mode, but at that point the device is typically reset and the error is logged. Figure 26-1 shows an overview of this level of operation.

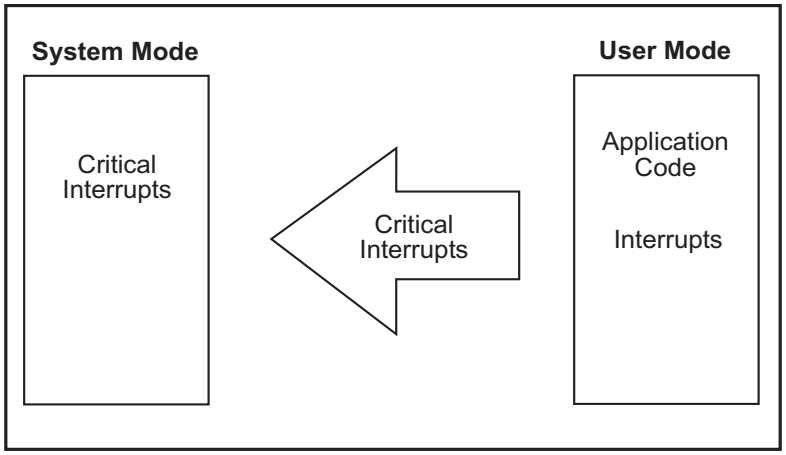


Figure 26-1. System/User Mode Setup for Memory Protection Only

26.3.2 Mixed System/User Mode Operation

This mode is similar to the previous mode, but with some portions of the program written for System Mode — for example, peripheral interrupts where latency is critical. By keeping the System Mode code sections small, potential system crashes are still minimized. Figure 26-2 shows an overview of this level of operation.

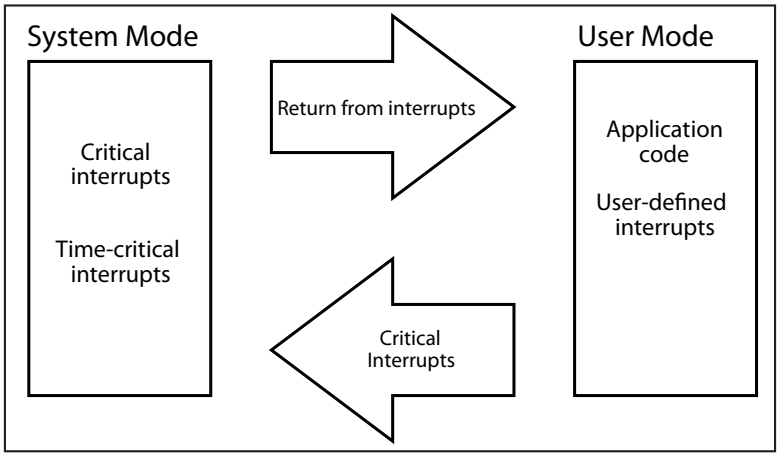


Figure 26-2. System/User Mode Setup for Mixed Operation

26.3.3 Complete Operating System

This section describes a "full" use of the System/User Mode — separating all common functions into a System Mode "operating system" while letting the application-specific code run in the User Mode. By default, the System Mode handles all peripherals and interrupts, as well as high-level interfaces such as a flash file system. However, the processor will be running the application code in the User Mode most of the time.

The application code can request direct access to a peripheral and/or interrupt from the System Mode. If allowed, the System Mode can create an interrupt vector as described in Section 26.3.7 that will execute the user code interrupt handler.

When the application code wants to perform an action that is controlled by the System Mode, it can request the particular action by loading the appropriate value into **HL** and executing **SYSCALL**. This requires generating a list of all the actions that the application code would want to do, assigning values to each action, and implementing a **SYSCALL** handler in the System Mode that parses the value passed to it and calls the appropriate function.

Write protection should be enabled (User Mode only) for all blocks containing system code and data as well as any critical memory regions.

If any critical interrupts occur (stack limit violation, system mode violation, write protection violation), System Mode handlers can perform any of a number of operations: restart the application code, signal another device, halt operation, and so on.

Figure 26-3 shows an overview of this level of operation.

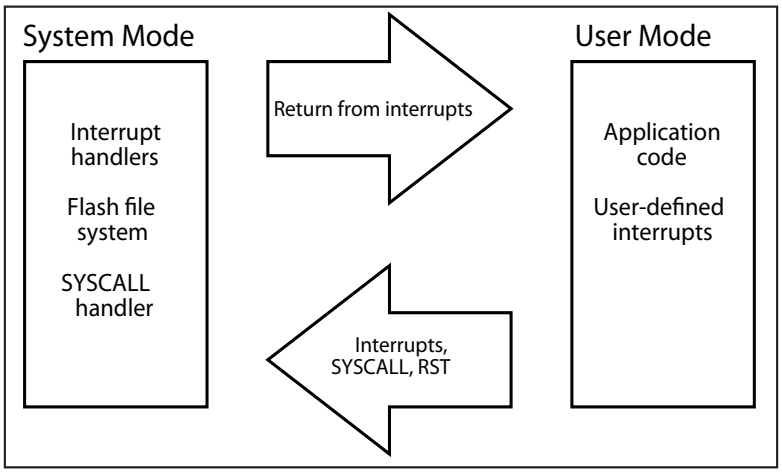


Figure 26-3. System/User Mode Setup for Operating System

26.3.4 Enabling the System/User Mode

The following steps describe how to enable the System/User Mode.

1. If a peripheral needs to be accessed while in User Mode, write to the appropriate user enable register to allow that access.
2. Write a 1 to bit 0 of EDMR to enable System/User Mode.
3. Execute the **`SETUSR`** instruction to enter User Mode.

After the User Mode is entered, the limitations described earlier are in effect — writes to protected registers will be ignored, Priority 3 is not available, and executing an **`IDET`** will cause a System Mode Violation interrupt. Other features such as write protection may be in effect for the User Mode as well.

26.3.5 System/User Mode Instructions

Seven new opcodes have been added to support the System/User mode, and are listed in Table 26-3. All but **IDET** are placed in previously empty opcode table assignments. **IDET** shares the value of **LD E,E** in the opcode table, and will perform that operation when the System/User mode is disabled, or when it is enabled and in the System mode. In addition, if the **ALTD** prefix appears before the opcode, **LD E',E** is always executed instead.

The processor keeps a one-byte stack (called the SU register) that is analogous to the IP register that keeps track of the interrupt priority. Every time **SETUSR** is executed (to enter the User mode), or an interrupt occurs, or **SYSCALL** or **RST** is executed (to enter the System mode), the current mode is pushed onto the SU register. When a **SURES** is executed, the previous mode is popped off the SU register.

The effects of each opcode are:

- The **SETUSR** opcode puts the processor into the User mode by pushing the correct value into the SU register.
- **PUSH SU** and **POP SU** push and pop the single-byte SU register on/off the SP stack.
- **SURES** pops the current processor mode off the SU register, returning it to the previous mode.
- **IDET** causes an interrupt if executed in the User mode, and does nothing in the System mode.
- **RDMODE** returns the current mode in the carry flag (0 for System mode, 1 for User mode).
- **SYSCALL** is essentially a new **RST** opcode, and was added to allow User mode access to the System mode without using one of the existing **RST** opcodes. It will put the processor into the System mode and execute code in the corresponding interrupt-vector table entry.

Table 26-3. New System/User Mode Opcodes

Instruction	Bytes	clk	A	I	S	Z	V	C	Operation	Priv ?
SETUSR	2	4		-	-	-	-	-	SU = {SU[5:0], 0x01}	Yes
PUSH SU	2	9		-	-	-	-	-	(SP-1) = SU; SP = SP - 1	Yes
POP SU	2	7		-	-	-	-	-	SU = (SP); SP = SP + 1	Yes
SURES	2	4		-	-	-	-	-	SU = {SU[1:0], SU[7:2]}	Yes
IDET	1	2		-	-	-	-	-	Performs **LD E,E**, but if (EDMF && SU[0]) then the System Violation interrupt flag is set; if ALTD appears before it always does **LD E',E**	No
RDMODE	2	4		-	-	-	-	*	CF = SU[0]	Yes
SYSCALL	2	10		-	-	-	-	-	SP = SP - 2; PC = {R,v} where v = SYSCALL offset	No

26.3.6 System Mode Violation Interrupt

The following steps describe how to set up the System Mode Violation interrupt.

1. Write the vector to the interrupt service routine to the internal interrupt table.
2. Enable the system/user mode by writing to EDMR.
3. The interrupt request is cleared automatically when handled.

A sample interrupt handler is shown below.

```
sysmode_isr::
   push af
   ; handle the system mode violation here
   pop af
   sures
   ipres
   ret
```

26.3.7 Handling Interrupts in the System/User Mode

Interrupts, **RST**s, **SYSCALL**, and **SCALL** all enter the System Mode automatically. There will be times, however, that an interrupt should be handled in the User Mode. The solution to this is for System Mode interrupt vector to reenter the User Mode before calling the User Mode interrupt handler. An example of both system and user interrupt handling is shown in Figure 26-4.

When enabled for User Mode access, a peripheral interrupt (if it is capable of generating an interrupt) can only be requested at Priority 2 or 1.

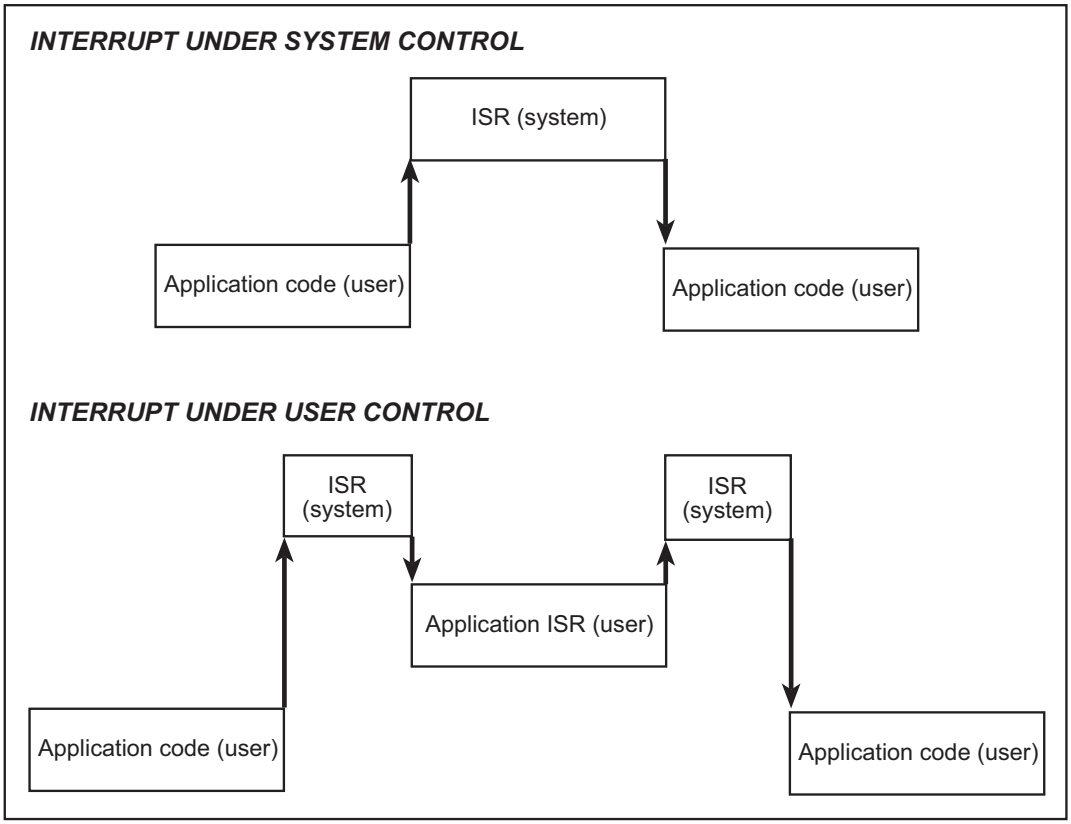


Figure 26-4. Interrupt Handing in the System/User Mode

Some sample code for both System Mode interrupts and User Mode interrupts is shown below. The use of **SETUSRP** and **SETSYSP** provides checks against stack mismatches and incorrect System/User Modes coming out of the User Mode handler.

```
systemmode_isr:                 ; jumped to from interrupt vector table
    ... handle interrupt ...
   sures                        ; reenter previous mode
   ipres                        ; restore previous interrupt priority
   ret

usermode_isr:                   ; jumped to from interrupt vector table
                                ;    (still in system mode at this point)
   push su                      ; preserve current SU stack
   setusrp 0x1234               ; enter user mode with stack compare value
   call user_handler            ; handle interrupt at user level
   setsysp 0x1234               ; return to system mode
   sures                        ; reenter previous mode
   ipres                        ; restore previous interrupt priority
   ret
```

26.4 Register Descriptions

Enable Dual-Mode Register (EDMR) (Address = 0x0420)		
Bit(s)	**Value**	**Description**
7:1		These bits are reserved and should be written with zeros.
0	0	Normal (System Mode only) operation.
	1	Enable System Mode/User Mode operation.

Real-Time Clock User Enable Register (RTUER) (Address = 0x0300)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to the RTC (I/O addresses 0x0002–0x0007).
	1	Enable User Mode access to the RTC (I/O addresses 0x0002–0x0007).
6:0		These bits are reserved and should be written with zeros.

Slave Port User Enable Register (SPUER) (Address = 0x0320)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to the slave port (I/O addresses 0x0020–0x0027).
	1	Enable User Mode access to the slave port (I/O addresses 0x0020–0x0027).
6:0		These bits are reserved and should be written with zeros.

Parallel Port A User Enable Register (PAUER) (Address = 0x0330)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port A (I/O addresses 0x0030–0x0037).
	1	Enable User Mode access to Parallel Port A (I/O addresses 0x0030–0x0037).
6:0		These bits are reserved and should be written with zeros.

Parallel Port B User Enable Register (PBUER) (Address = 0x0340)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port B (I/O addresses 0x0040–0x0047).
	1	Enable User Mode access to Parallel Port B (I/O addresses 0x0040–0x0047).
6:0		These bits are reserved and should be written with zeros.

Parallel Port C User Enable Register (PCUER) (Address = 0x0350)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port C (I/O addresses 0x0050–0x0055).
	1	Enable User Mode access to Parallel Port C (I/O addresses 0x0050–0x0055).
6:0		These bits are reserved and should be written with zeros.

Parallel Port D User Enable Register (PDUER) (Address = 0x0360)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port D (I/O addresses 0x0060–0x006F).
	1	Enable User Mode access to Parallel Port D (I/O addresses 0x0060–0x006F).
6:0		These bits are reserved and should be written with zeros.

Parallel Port E User Enable Register (PEUER) (Address = 0x0370)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port E (I/O addresses 0x0070–0x007F).
	1	Enable User Mode access to Parallel Port E (I/O addresses 0x0070–0x007F).
6:0		These bits are reserved and should be written with zeros.

Parallel Port F User Enable Register (PFUER) (Address = 0x0338)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port E (I/O addresses 0x0038–0x003F).
	1	Enable User Mode access to Parallel Port E (I/O addresses 0x0038–0x003F).
6:0		These bits are reserved and should be written with zeros.

Parallel Port G User Enable Register (PGUER) (Address = 0x0348)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Parallel Port E (I/O addresses 0x0048–0x004F).
	1	Enable User Mode access to Parallel Port E (I/O addresses 0x0048–0x004F).
6:0		These bits are reserved and should be written with zeros.

Input Capture User Enable Register (ICUER) (Address = 0x0358)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Input Capture (I/O addresses 0x0056–0x005F).
	1	Enable User Mode access to Input Capture (I/O addresses 0x0056–0x005F).
6:0		These bits are reserved and should be written with zeros.

I/O Bank User Enable Register (IBUER) (Address = 0x0380)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to I/O Bank 7 (and internal I/O address 0x0087).
	1	Enable User Mode access to I/O Bank 7 (and internal I/O addresses 0x0087).
6	0	Disable User Mode access to I/O Bank 6 (and internal I/O address 0x0086).
	1	Enable User Mode access to I/O Bank 6 (and internal I/O addresses 0x0086).
5	0	Disable User Mode access to I/O Bank 5 (and internal I/O address 0x0085).
	1	Enable User Mode access to I/O Bank 5 (and internal I/O addresses 0x0085).
4	0	Disable User Mode access to I/O Bank 4 (and internal I/O address 0x0084).
	1	Enable User Mode access to I/O Bank 4 (and internal I/O addresses 0x0084).
3	0	Disable User Mode access to I/O Bank 3 (and internal I/O address 0x0083).
	1	Enable User Mode access to I/O Bank 3 (and internal I/O addresses 0x0083).
2	0	Disable User Mode access to I/O Bank 2 (and internal I/O address 0x0082).
	1	Enable User Mode access to I/O Bank 2 (and internal I/O addresses 0x0082).
1	0	Disable User Mode access to I/O Bank 1 (and internal I/O address 0x0081).
	1	Enable User Mode access to I/O Bank 1 (and internal I/O addresses 0x0081).
0	0	Disable User Mode access to I/O Bank 0 (and internal I/O address 0x0080).
	1	Enable User Mode access to I/O Bank 0 (and internal I/O addresses 0x0080).

PWM User Enable Register (PWUER) (Address = 0x0388)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to the Pulse Width Modulator (I/O addresses 0x0088 to 0x008F).
	1	Enable User Mode access to the Pulse Width Modulator (I/O addresses 0x0088 to 0x008F).
6:0		These bits are reserved and should be written with zeros.

Quad Decode User Enable Register (QDUER) (Address = 0x0390)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to the Quadrature Decoder (I/O addresses 0x0090–0x0097).
	1	Enable User Mode access to the Quadrature Decoder (I/O addresses 0x0090–0x0097).
6:0		These bits are reserved and should be written with zeros.

External Interrupt User Enable Register (IUER) (Address = 0x0398)		
Bit(s)	**Value**	**Description**
7:2		These bits are reserved and should be written with zeros.
1	0	Disable User Mode access to External Interrupt 1 (I/O address 0x0099).
	1	Enable User Mode access to External Interrupt 1 (I/O addresses 0x0099).
0	0	Disable User Mode access to External Interrupt 0 (I/O address 0x0098).
	1	Enable User Mode access to External Interrupt 0 (I/O addresses 0x0098).

Timer A User Enable Register (TAUER) (Address = 0x03A0)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Timer A (I/O addresses 0x00A0–0x00AF).
	1	Enable User Mode access to Timer A (I/O addresses 0x00A0–0x00AF).
6:0		These bits are reserved and should be written with zeros.

Timer B User Enable Register (TBUER) (Address = 0x03B0)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Timer B (I/O addresses 0x00B0–0x00BF).
	1	Enable User Mode access to Timer B (I/O addresses 0x00B0–0x00BF).
6:0		These bits are reserved and should be written with zeros.

Serial Port A User Enable Register (SAUER) (Address = 0x03C0)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Serial Port A (I/O addresses 0x00C0–0x00C7).
	1	Enable User Mode access to Serial Port A (I/O addresses 0x00C0–0x00C7).
6:0		These bits are reserved and should be written with zeros.

Serial Port B User Enable Register (SBUER) (Address = 0x03D0)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Serial Port B (I/O addresses 0x00D0–0x00D7).
	1	Enable User Mode access to Serial Port B (I/O addresses 0x00D0–0x00D7).
6:0		These bits are reserved and should be written with zeros.

Serial Port C User Enable Register (SCUER) (Address = 0x03E0)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Serial Port C (I/O addresses 0x00E0–0x00E7).
	1	Enable User Mode access to Serial Port C (I/O addresses 0x00E0–0x00E7).
6:0		These bits are reserved and should be written with zeros.

Serial Port D User Enable Register (SDUER) (Address = 0x03F0)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Serial Port D (I/O addresses 0x00F0–0x00F7).
	1	Enable User Mode access to Serial Port D (I/O addresses 0x00F0–0x00F7).
6:0		These bits are reserved and should be written with zeros.

Serial Port E User Enable Register (SEUER) (Address = 0x03C8)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Serial Port E (I/O addresses 0x00C8–0x00CF).
	1	Enable User Mode access to Serial Port E (I/O addresses 0x00C8–0x00CF).
6:0		These bits are reserved and should be written with zeros.

Serial Port F User Enable Register (SFUER) (Address = 0x03D8)		
Bit(s)	**Value**	**Description**
7	0	Disable User Mode access to Serial Port F (I/O addresses 0x00D8–0x00DF).
	1	Enable User Mode access to Serial Port F (I/O addresses 0x00D8–0x00DF).
6:0		These bits are reserved and should be written with zeros.

27. SPECIFICATIONS

27.1 DC Characteristics

Table 27-1. DC Electrical Characteristics

Parameter	Symbol	Min	Typ	Max	Units
Operating Temperature	T_A	-55		85	°C
Storage Temperature	T_S	-65		150	°C
Supply Voltage	V_{DD}	3.0	3.3	3.6	V
Maximum Input Voltage: • Oscillator Buffer Input • 5-V-tolerant I/O				 V_{DD} + 0.5 5.5	V
High-Level Input Voltage	V_{IH}	2.0			V
Low-Level Input Voltage	V_{IL}			0.8	V
High-Level Output Voltage	V_{OH}	0.7 × V_{DD}			V
Low-Level Output Voltage	V_{OL}			0.4	V
High-Level Input Current (absolute worst case, all buffers)	I_{IH}			10	µA
Low-Level Input Current (absolute worst case, all buffers)	I_{IL}	-10			µA
High-Impedance State Output Current (absolute worst case, all buffers)	I_{OZ}	-10		10	µA

Table 27-2. Preliminary Battery-Backed DC Electrical Characteristics (V_{DD} = 3.3V ± 10%, T_A = -55°C to 85°C)

Parameter	Symbol	Min	Typ	Max
VBAT Supply Voltage (device powered) (device powered down)	VBAT	 1.65 V 1.65 V	 3.3 V 1.8 V	 3.6 V 3.6 V
VBAT Current (device powered down)	I_{VBAT}		7 µA @ 3.3 V 1 µA @ 1.8 V	

27.2 AC Characteristics

Table 27-3. Preliminary AC Electrical Characteristics (V_{DD} = 3.3 V ± 10%, T_A = -55°C to 85°C)

Parameter	Symbol	Min	Typ	Max
Main Clock Frequency on CLKI	f_{main}			58.8 MHz
Real-Time Clock Frequency on CLK32K	f_{RTC}		32.768 kHz	

27.3 Memory Access Times

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

27.3.1 Memory Reads

Table 27-4. Preliminary Memory Read Time Delays (V_{DD} = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Loading	Min	Typ	Max
Clock to Address Delay	T_{adr}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to Memory Chip Select Delay	T_{CSx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to Memory Read Strobe Delay	T_{OEx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Data Setup Time	T_{setup}	-		1 ns	
Data Hold Time	T_{hold}	-		0 ns	

27.3.2 Memory Writes

Table 27-5. Preliminary Memory Write-Time Delays (V_{DD} = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Loading	Min	Typ	Max
Clock to Address Delay	T_{adr}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to Memory Chip Select Delay	T_{CSx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to Memory Write Strobe Delay	T_{WEx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
High Z to Data Valid Relative to Clock	T_{DHZV}	30 pF		10 ns	
		60 pF		12 ns	
		90 pF		15 ns	
Data Valid to High Z Relative to Clock	T_{DVHZ}	30 pF		10 ns	
		60 pF		12 ns	
		90 pF		15 ns	

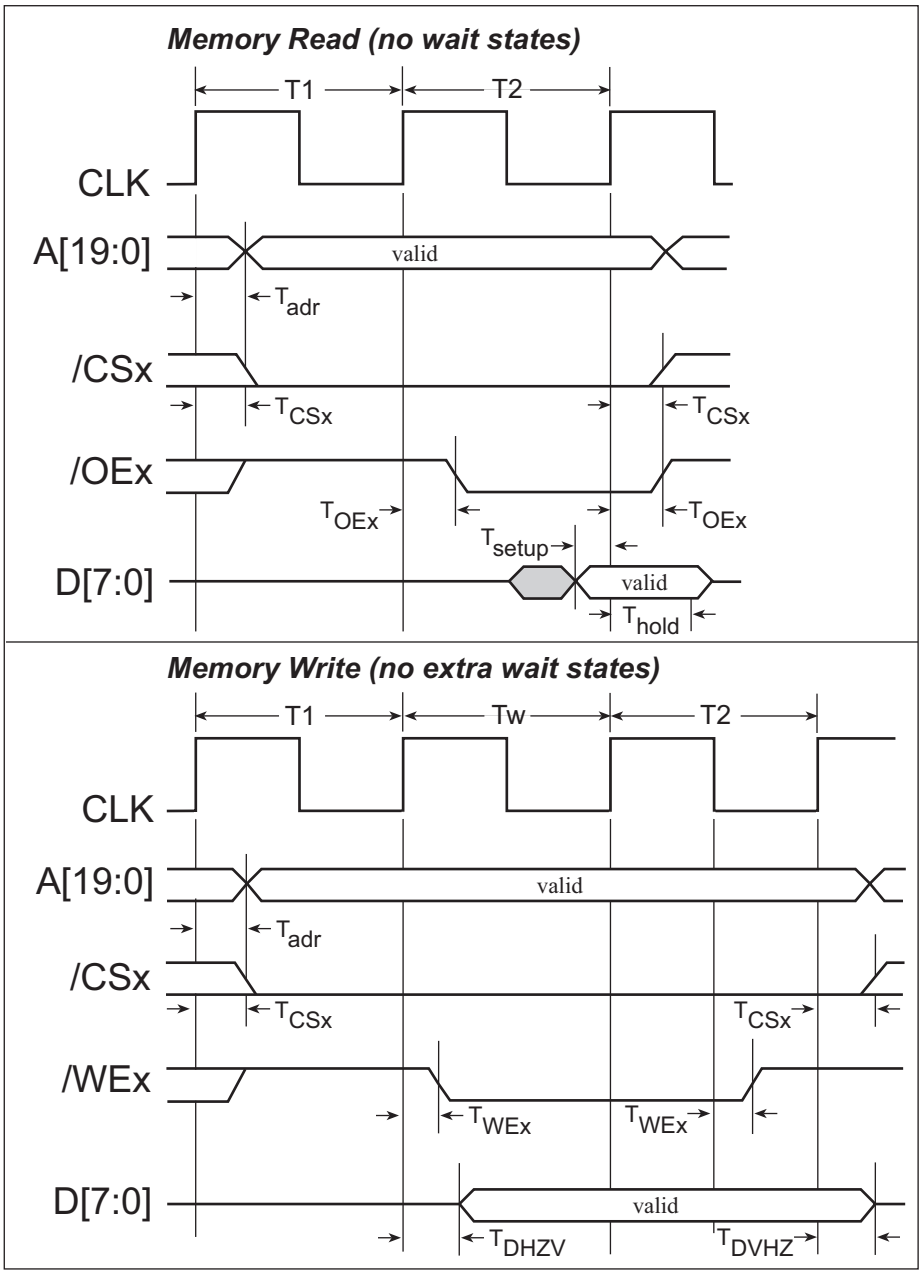


Figure 27-1. Memory Read and Write Cycles

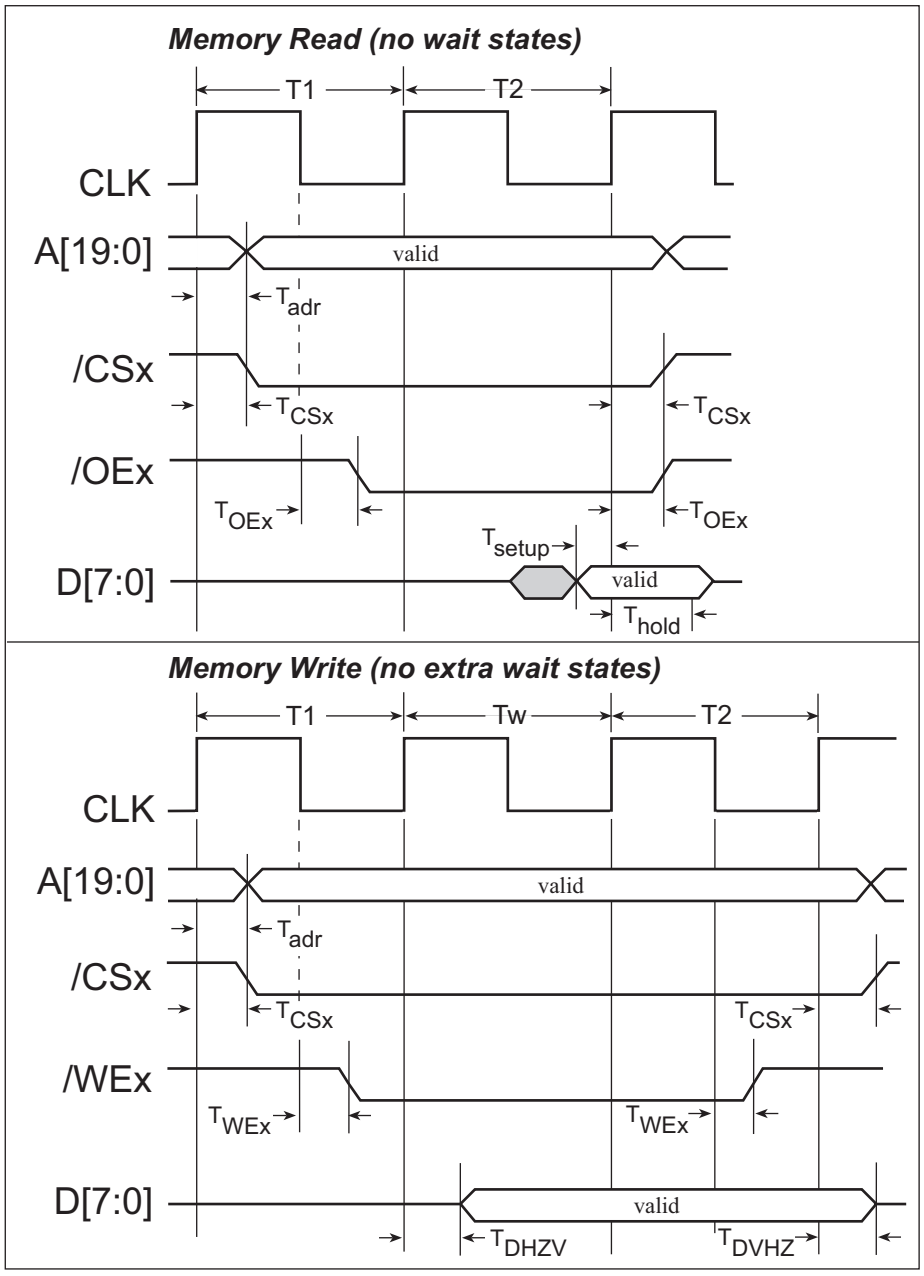


Figure 27-2. Memory Read and Write Cycles—Early Output Enable and Write Enable Timing

27.3.3 External I/O Reads

Table 27-6. Preliminary External I/O Read Time Delays (V_{DD} = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Loading	Min	Typ	Max
Clock to Address Delay	T_{adr}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to Memory Chip Select Delay	T_{CSx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to I/O Chip Select Delay	T_{IOCSx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to I/O Read Strobe Delay	T_{IORD}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to I/O Buffer Enable Delay	T_{BUFEN}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Data Setup Time	T_{setup}	-		1 ns	
Data Hold Time	T_{hold}	-		0 ns	

27.3.4 External I/O Writes

Table 27-7. Preliminary External I/O Write Time Delays
(V_{DD} = 3.3 V ± 10%, T_A = -40°C to 85°C)

Parameter	Symbol	Loading	Min	Typ	Max
Clock to Address Delay	T_{adr}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to Memory Chip Select Delay	T_{CSx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to I/O Chip Select Delay	T_{IOCSx}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to I/O Write Strobe Delay	T_{IOWR}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
Clock to I/O Buffer Enable Delay	T_{BUFEN}	30 pF		6 ns	
		60 pF		8 ns	
		90 pF		11 ns	
High Z to Data Valid Relative to Clock	T_{DVHZ}	30 pF		10 ns	
		60 pF		12 ns	
		90 pF		15 ns	
Data Valid to High Z Relative to Clock	T_{DVHZ}	30 pF		10 ns	
		60 pF		12 ns	
		90 pF		15 ns	

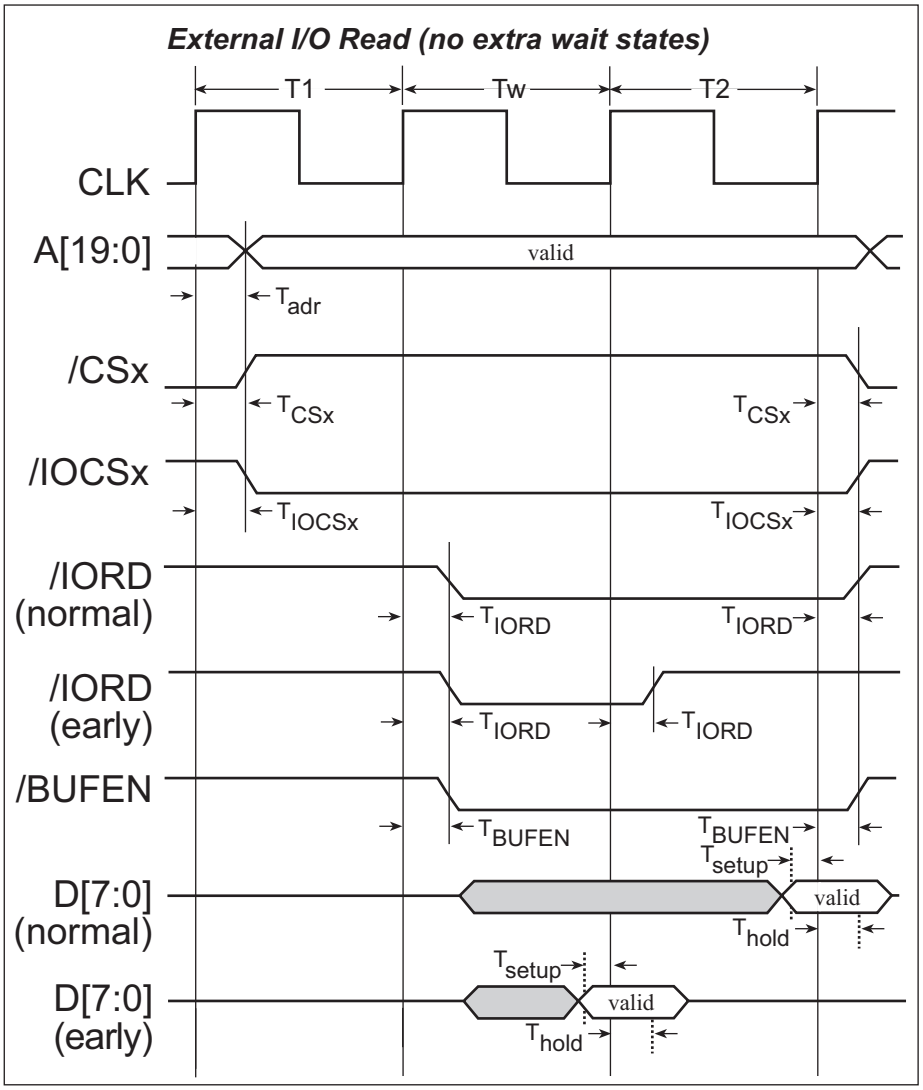


Figure 27-3. I/O Read Cycles—No Extra Wait States

NOTE: The early /IORD and data timing results are not applicable to the original Rabbit 3000 microprocessor. They apply starting with the Rabbit 3000A.

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

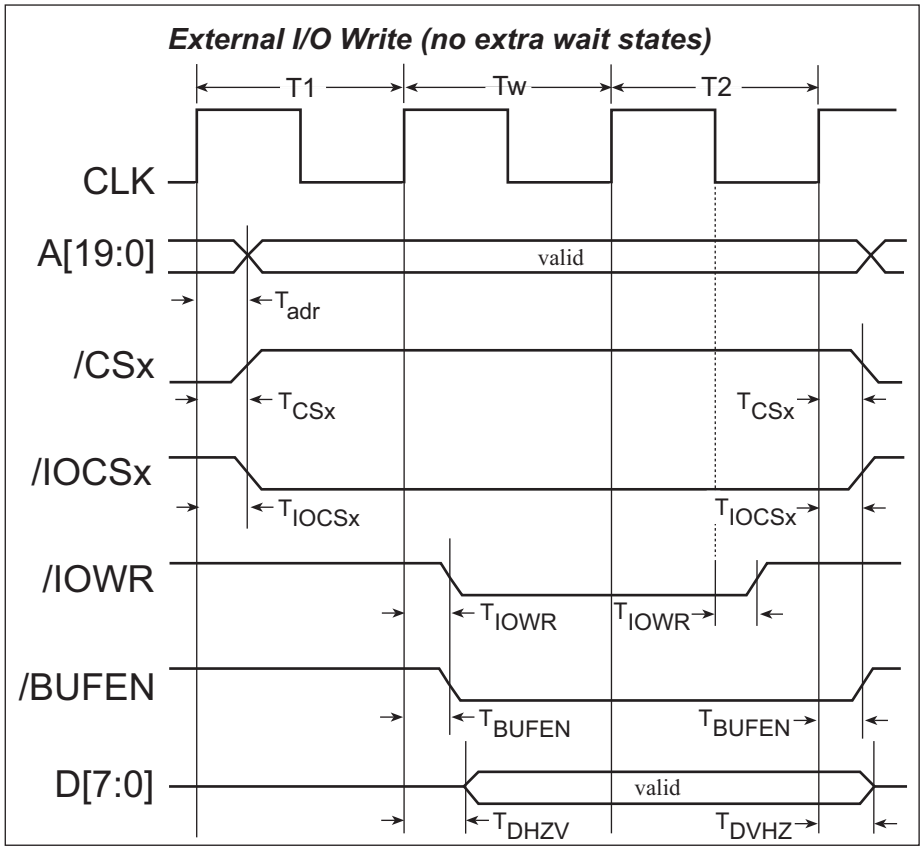


Figure 27-4. I/O Write Cycles—No Extra Wait States

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

27.3.5 Memory Access Times

In computing memory requirements, the important considerations are the address access time, output-enable access time, and minimum write-pulse required. Increasing the clock doubler delay increases the output-enable time, but decreases the memory write-pulse width. The early write-pulse option can be used to ensure a long-enough write pulse, but then it must be ensured that the write pulse does not begin before the address lines have stabilized.

The clock doubler has an affect on the memory access times. It works by ORing the clock with a delayed version of itself. The nominal delay varies from 3 to 20 ns, and is set under program control. Any asymmetry in the main clock input before it is doubled will result in alternate clocks having slightly different periods. Using the suggested oscillator circuit, the asymmetry is no worse than 52%–48%. This results in a given clock being shortened by the ratio 50/52, or 4% worst-case. The memory access time is not normally affected because the memory bus cycle is two clocks long and includes both a long and a short clock, resulting in no net change arising from asymmetry. However, if an odd number of wait states is used, then the memory access time will be affected slightly.

When the clock spectrum spreader is enabled, clock periods are shortened by a small amount, depending on whether the "normal" or the "strong" spreader setting is used, and depending on the operating voltage. If the clock doubler is used, the spectrum spreader affects every other cycle and reduces the clock high time. If the doubler is not used, then the spreader affects every clock cycle, and the clock low time is reduced. Of course, the spectrum spreader also lengthens clock cycles, but only the worst-case shortening is relevant for calculating worst-case access times. The numbers given for clock shortening with the doubler disabled are the combined shortening for two consecutive clock cycles, worst case.

The required memory address and output-enable access time for some typical clock speeds are given in Table 27-8 below. It is assumed that the clock doubler is used, that the clock spreader is enabled in the normal mode, that the memory early output-enable is on, and that the address bus has a load of 60 pF.

Table 27-8. Preliminary Memory Requirements
(V_{DD} = 3.3 V ± 10%, T_A = -40°C to 85°C, address bus loading = 60 pF)

Clock Frequency (MHz)	Period (ns)	Clock Doubler Nominal Delay (ns)	Memory Address Access (ns)	Memory Output-Enable Access (ns)
18.43	54	20	97	60
22.11	45	20	78	51
24.00	42	19	72	45
25.80	39	17	66	43
29.49	34	16	56	37
44.24	22.5	10	33.5	22

All important signals on the Rabbit 3000 are output-synchronized with the internal clock. The internal clock is closely synchronized with the external clock, which is available on the CLK pin. The delay in signal output depends on the capacitive load on the output lines. In the case of the address lines, which are critically important for establishing memory access time requirements, the capacitive loading is usually in the range of 25–100 pF, and the load is due to the input capacitance of the memory devices and PC trace capacitance.

Delays are expressed from the waveform midpoint in keeping with the convention used by memory manufacturers.

Table 27-9 lists the delays in gross memory access time for several values of V_{DD}.

Table 27-9. Preliminary Data and Clock Delays (V_{DD} ±10%, Temp. -40°C to 85°C)

V_{DD} (V)	Clock to Address Output Delay (ns)			Data Setup Time Delay (ns)	Worst-Case Spectrum Spreader Delay (ns)		
	30 pF	60 pF	90 pF		0.5 ns setting no dbl / dbl	1 ns setting no dbl / dbl	2 ns setting no dbl / dbl
3.3	6	8	11	1	3/4.5	4.5/9	3.3
2.7	7	10	13	1.5	3.5/5.5	5.5/11	2.7
2.5	8	11	15	1.5	4/6	6/12	2.5
1.8	18	24	33	3	8/12	11/22	1.8

When the spectrum spreader is enabled with the clock doubler, every other clock cycle is shortened or lengthened by a maximum amount given in the table above. The shortening takes place by shortening the high part of the clock. If the doubler is not enabled, then every clock is shortened during the low part of the clock period. The maximum shortening for a pair of clocks combined is shown in the table.

The gross memory access time is 2T, where T is the clock period. To calculate the actual memory access time, subtract the clock to address output time, the data in setup time, and the clock period shortening due to the clock spectrum spreader from 2T.

Example Memory Access Time Calculation

- clock = 29.49 MHz, so T = 34 ns
- operating voltage is 3.3 V
- bus loading is 60 pF
- clock to address output delay = 8 ns (see Table 27-9)
- data setup time = 1 ns
- spectrum spreader is on in 1 ns mode, resulting in a loss of 3 ns worst-case (see Table 27-9)

The access time is given by

access time = 2T - (clock to address) - (data setup) - (spreader delay)
= 68 ns - 8 ns - 1 ns - 3 ns
= 56 ns

Similarly, the gross output-enable access time is T + minimum clock low time (it is assumed that the early output enable option is enabled). This is reduced by the spectrum spreader loss, the time from clock to output for the output enable signal, the data setup time, and a correction for the asymmetry of the original oscillator clock.

Example Output-Enable Access Time Calculation

NOTE: There is some process and temperature variation in the clock doubler settings. As a rule of thumb, a 20% variation should be considered. When the doubler is enabled, 80% of the nominal value should be used for the memory access time calculation.

- clock = 29.49 MHz, so T = 34 ns
- operating voltage is 3.3 V
- the clock doubler has a nominal delay of 16 ns (see Table 27-8), resulting in a minimum clock low time of 80% × 16 ns = 12.8 ns
- clock to output enable is 5 ns (assuming 20 pF load)
- spectrum spreader is on in normal mode, resulting in a loss of 3 ns worst-case (see Table 27-9)
- main clock asymmetry is 52% / 48%, resulting in a loss of 4% of the clock period, or 1.4 ns

The output enable access time is given by

access time = T + (min. clock low) - (clock to output enable) -
(spreader delay) - (asymmetry delay) - (data setup time)
= 34 ns + 12.8 ns - 5 ns - 3 ns - 1.4 ns - 1 ns
= 36 ns

27.4 Clock Speeds

27.4.1 Recommended Clock/Memory Configurations

The preferred configuration for a Rabbit-based system is to use an external crystal or resonator that has a frequency one-half of the maximum internal clock frequency. The oscillator frequency can be doubled or divided by 2, 4, 6, or 8, giving a variety of operating speeds from the same crystal frequency. In addition, the 32.768 kHz oscillator that drives the battery-backable clock can be used as the main processor clock and, to save the substantial power consumed by the fast oscillator, the fast oscillator can be turned off. This scenario is called the *sleepy mode*, where the clock speed is from 2 kHz to 32 kHz, and the operating system current consumption of 10 to 120 µA depends on frequency and voltage.

Table 27-10 describes some recommended clock and memory configurations for 8-bit memory devices. Optimal configurations for using 15 ns, 45–55 ns, and 70 ns memories are shown.

Table 27-10. Recommended Clock/Memory Configurations

Input Frequency (MHz)	Internal Frequency (MHz)	Recommended Memory Setup		Use
		SRAM	Flash	
29.4912	58.9824	8 bits, 15 ns, 0 wait states	8 bits, 45–55 ns, 2 wait states	Fastest 8-bit configuration without wait states (run code from SRAM)
22.1184	44.2368	8 bits, 15 ns, 0 wait states	8 bits, 45–55 ns, 1 wait state	Fastest 8-bit, 55 ns configuration with 1 wait state (run code in SRAM)
14.7456	29.4912	8 bits, 45–55 ns, 0 wait states	8 bits, 45–55 ns, 0 wait states	Fastest 8-bit, 55 ns configuration without wait states
11.0592	22.1184	8 bits, 70 ns, 0 wait states	8 bits, 70 ns, 0 wait states	Fastest 8-bit, 70 ns configuration without wait states

The Rabbit 3000 is rated for a minimum clock period of 16 ns for both commercial and industrial specifications (preliminary). The commercial rating calls for a ±5% voltage variation from 3.3 V, and a temperature range from -40 to + 70°C. The industrial ratings stretch the voltage variation to ±10% over a temperature range from -55 to + 85°C. This corresponds to maximum clock frequencies of about 58.8 MHz (industrial). If the clock doubler or spectrum spreader is used, these maximum ratings must be reduced as shown in Table 27-11.

Table 27-11. Maximum Clock Speeds at 3.3 V [Preliminary]

Conditions	Commercial Ratings		Industrial Ratings		Duty Cycle Requirements (ns)
	Minimum Period (ns)	Maximum Frequency (MHz)	Minimum Period (ns)	Maximum Frequency (MHz)	
No doubler or spreader	17	58.8	18	55.5	
Spreader only normal	20	50.0	21	47.6	
Spreader only strong	21	47.6	22	45.4	
Doubler only (8 ns delay)	19	52.6	20	50.0	1 > (clock low - clock high) > 0
Doubler only (internal 50% clock)	20	50	21	47.6	1 > (clock low - clock high) > -1
Spreader normal with doubler (8 ns delay)	21	47.6	22	45.4	4 > (clock low - clock high) > 2
Spreader normal with doubler (8 ns delay), internal 50% clock	24	41.6	25	40.0	1 > (clock low - clock high) > -1
Spreader only strong	21.5	46.5	22.5	45.0	
Spreader strong with doubler (8 ns delay)	23	43.5	24	41.6	8 > (clock low - clock high) > 6

When the doubler is used, the duty cycle of the clock becomes a critical parameter. The duty cycle should be measured at the separate clock output pin (pin 2). The minimum period must be increased by any amount that the clock high time is greater or less than specified in the duty-cycle requirement.

For example, consider a design where the spreader and doubler are enabled, with 8 ns nominal delay in the doubler. The high and low clock are equal to within 1 ns. This violates the duty cycle requirement by 3 ns since (clock low - clock high) can be as small as -1 ns, but the requirement is that it not be less than 2 ns. Thus, 3 ns must be added to the minimum period of 21 ns, giving a minimum period of 24 ns and a maximum frequency of 41.6 MHz (commercial).

Since the built-in high-speed oscillator buffer generates a clock that is very close to having a 50% duty cycle, to obtain the highest clock speeds using the clock doubler you must use an external oscillator buffer that will allow for duty-cycle adjustment by changing the resistance of the power and ground connections as shown below.

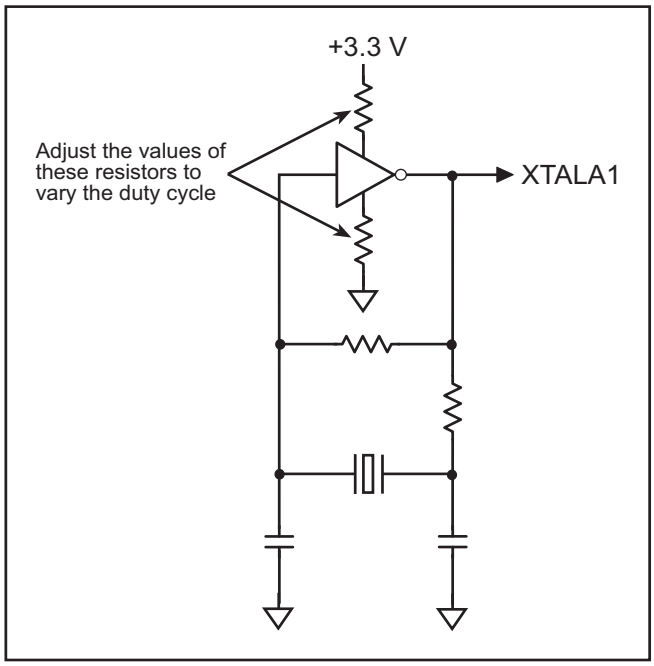


Figure 27-5. External Oscillator Buffer

27.5 Power and Current Consumption

Various mechanisms contribute to the current consumption of the Rabbit 3000 processor while it is operating, including current that is proportional to the voltage alone (leakage current) and dependent on both voltage and frequency (switching and crossover current). To reduce current consumption, the clock can be divided down in one of the sleepy modes; see Table 25-1 for more details.

Figure 27-6 shows a typical current draw as a function of the main clock frequency. The values shown do not include any current consumed by external oscillators or memory. It is assumed that approximately 30 pF is connected to each address line.

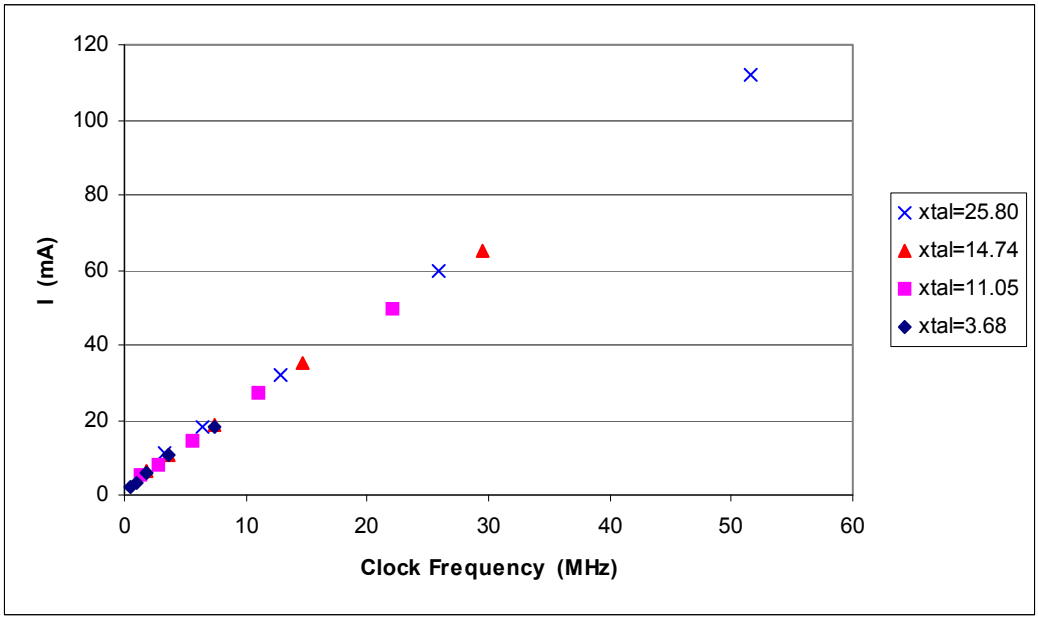


Figure 27-6. Typical Current Draw as a Function of the Main Clock Frequency at 3.3 V

27.5.1 Sleepy Mode Current Consumption

The Rabbit 3000 supports designs with very low power consumption by using features such as the sleepy modes and self-timed chip selects. At the low frequencies possible in the ultra-sleepy modes (as low as 2 kHz), the external memory devices become significant factors in the current consumption unless one of the short or self-timed chip selects are used.

Figure 27-7 shows a typical current draw for the sleepy modes.

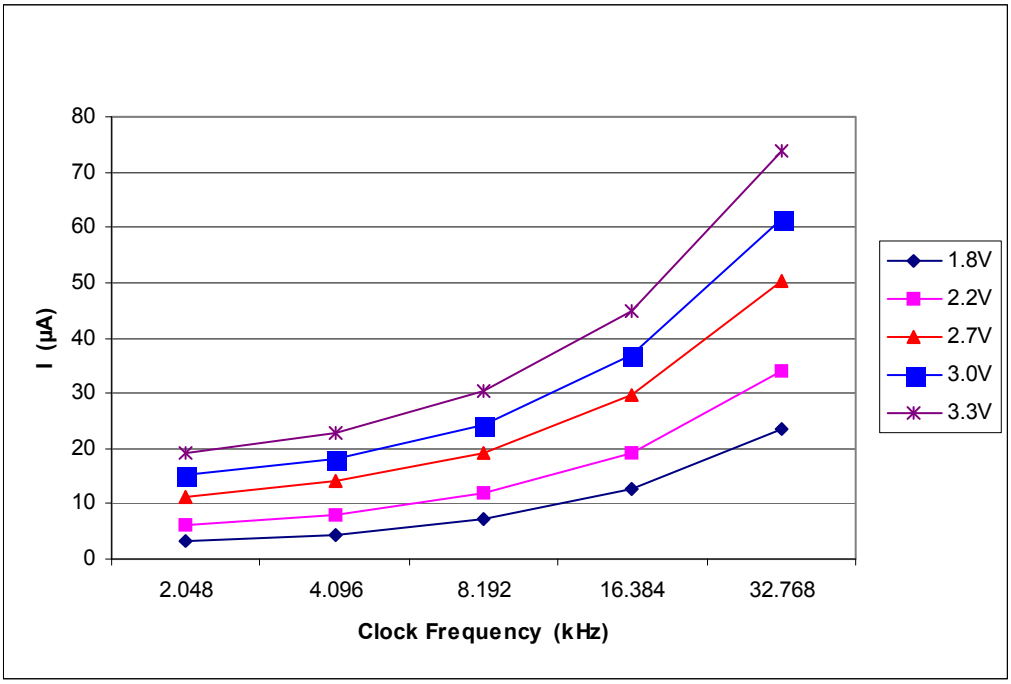


Figure 27-7. Typical Current Draw for the Sleepy Modes

27.5.2 Battery-Backed Clock Current Consumption

For the battery-backed features of the Rabbit 3000 to perform while the processor is powered down, both the VBAT and BATIO pins need to be supplied properly. The VBAT pin powers the internal real-time clock and the battery-backed SRAM, while VBATIO powers the /RESET, /CS1, CLK32K, and RESOUT pins.

Note that the VBATIO pin can be powered at 1.8 V during powerdown even if the processor is running at 3.3 V normally. A circuit to switch between a 1.8–2.0 V battery and the main power can use the RESOUT pin to switch the power source for the VBATIO pin. R is a current-limiting resistor that should be adjusted for the battery voltage; a good value to use for a 3.0 V battery is 150 kΩ.

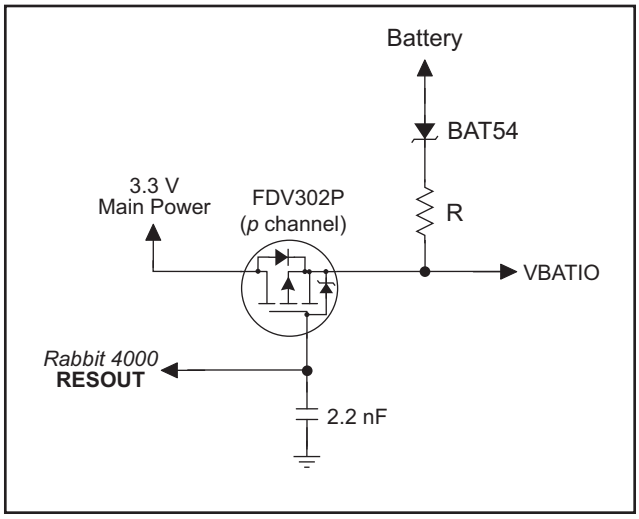


Figure 27-8. Switching Circuit for VATIO Pin

Table 27-12 shows the typical current consumption for these pins while the remainder of the Rabbit 3000 is powered down.

Table 27-12. Typical Battery-Backed Current Consumption (-40°C to +85°C)

Pin	Voltage	Current
VBAT	3.3 V	7 µA
	1.8 V	1 µA

27.6 Reduced-Power External Main Oscillator

The circuit in Figure 27-9 can be used to generate the main clock using less power than with the built-in oscillator buffer. The power consumption is less because of the current-limiting resistors that cannot be used with the built-in buffer. The 2.2 kΩ series resistor must be reduced as the clock frequency increases, as must be the current-limiting resistors.

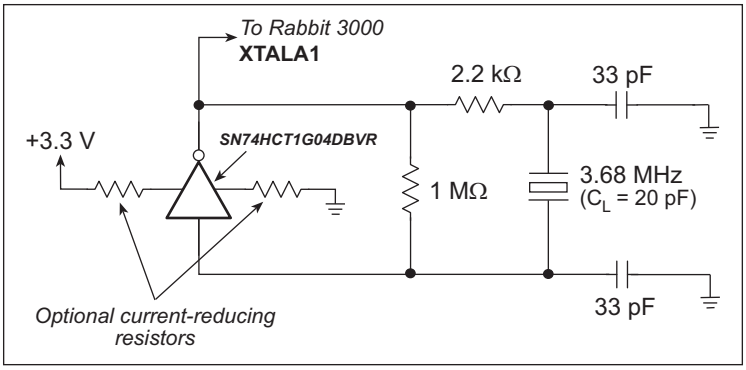


Figure 27-9. Reduced-Power External Main Oscillator

Table 27-13 lists results for the reduced-power external oscillator with no current-limiting resistors.

Table 27-13. Current Draw Using Reduced-Power External Oscillator (0 Ω current-limiting resistors)

Voltage (V)	Current (incl built-in buffer) (mA)
3.3	0.635
2.5	0.380
1.8	0.252

Design Recommendations

- Add current-limiting resistors to reduce current without inhibiting oscillator start-up
- Increase the 1 MΩ resistor to improve gain
- Minimize loop area to reduce EMI

28. PACKAGE SPECIFICATIONS AND PINOUT

28.1 LQFP Package

28.1.1 Pinout

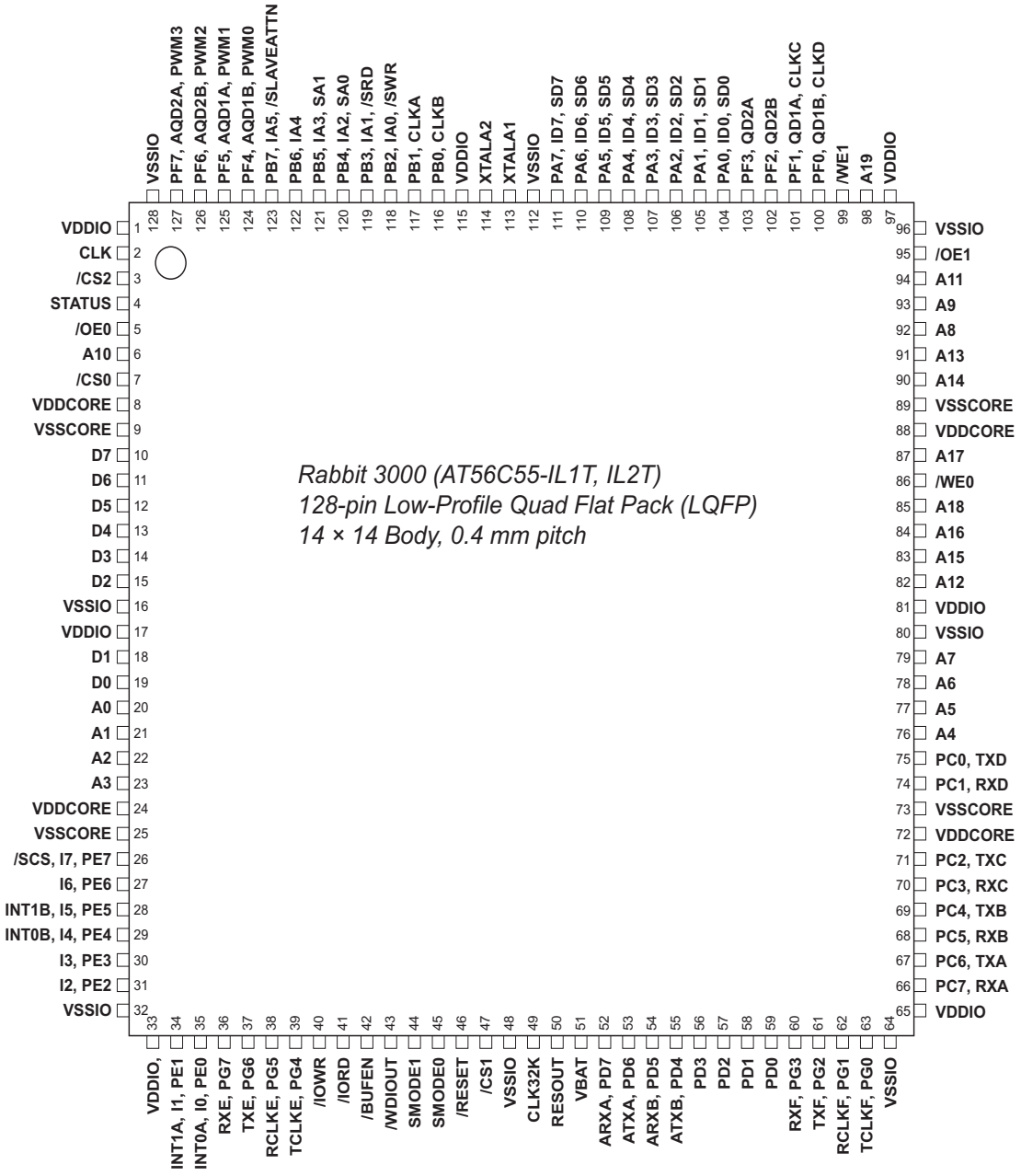


Figure 28-1. Package Outline and Pin Assignments

28.1.2 Mechanical Dimensions and Land Pattern

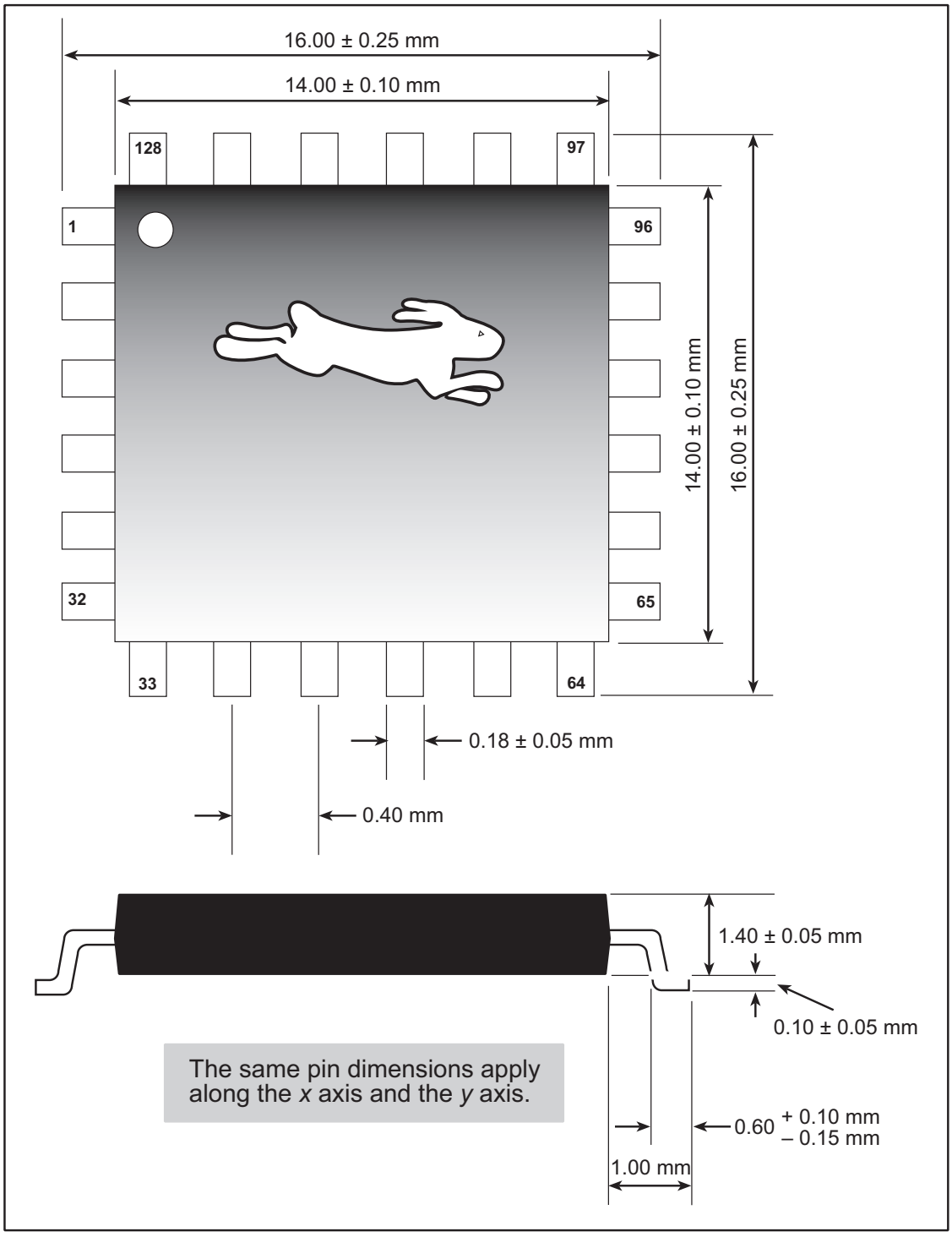


Figure 28-2. Mechanical Dimensions Rabbit LQFP Package

Figure 28-3 shows the PC board land pattern for the Rabbit 3000 chip in a 128-pin LQFP 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.

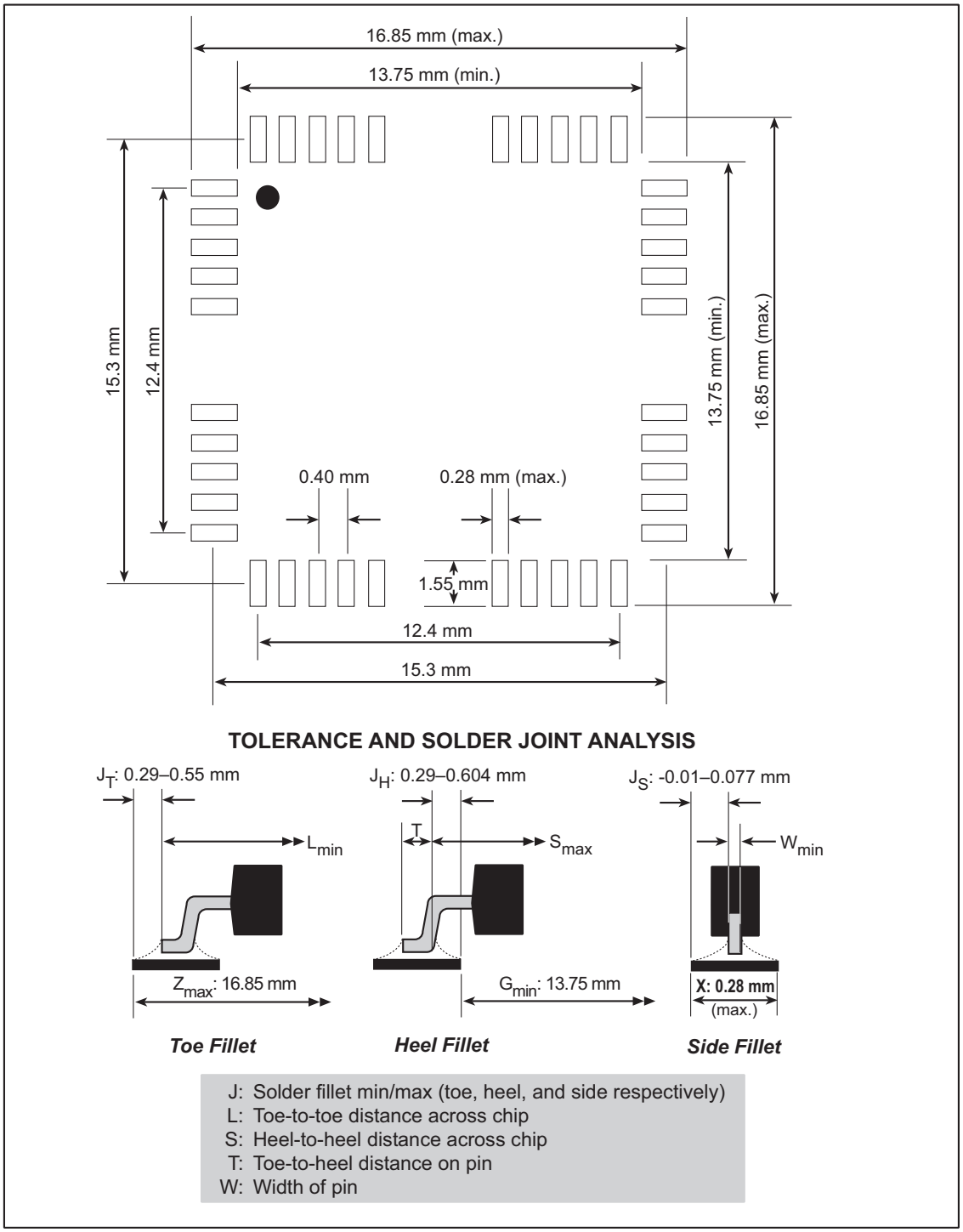


Figure 28-3. PC Board Land Pattern for Rabbit 4000 128-pin LQFP

28.2 Ball Grid Array Package

28.2.1 Pinout

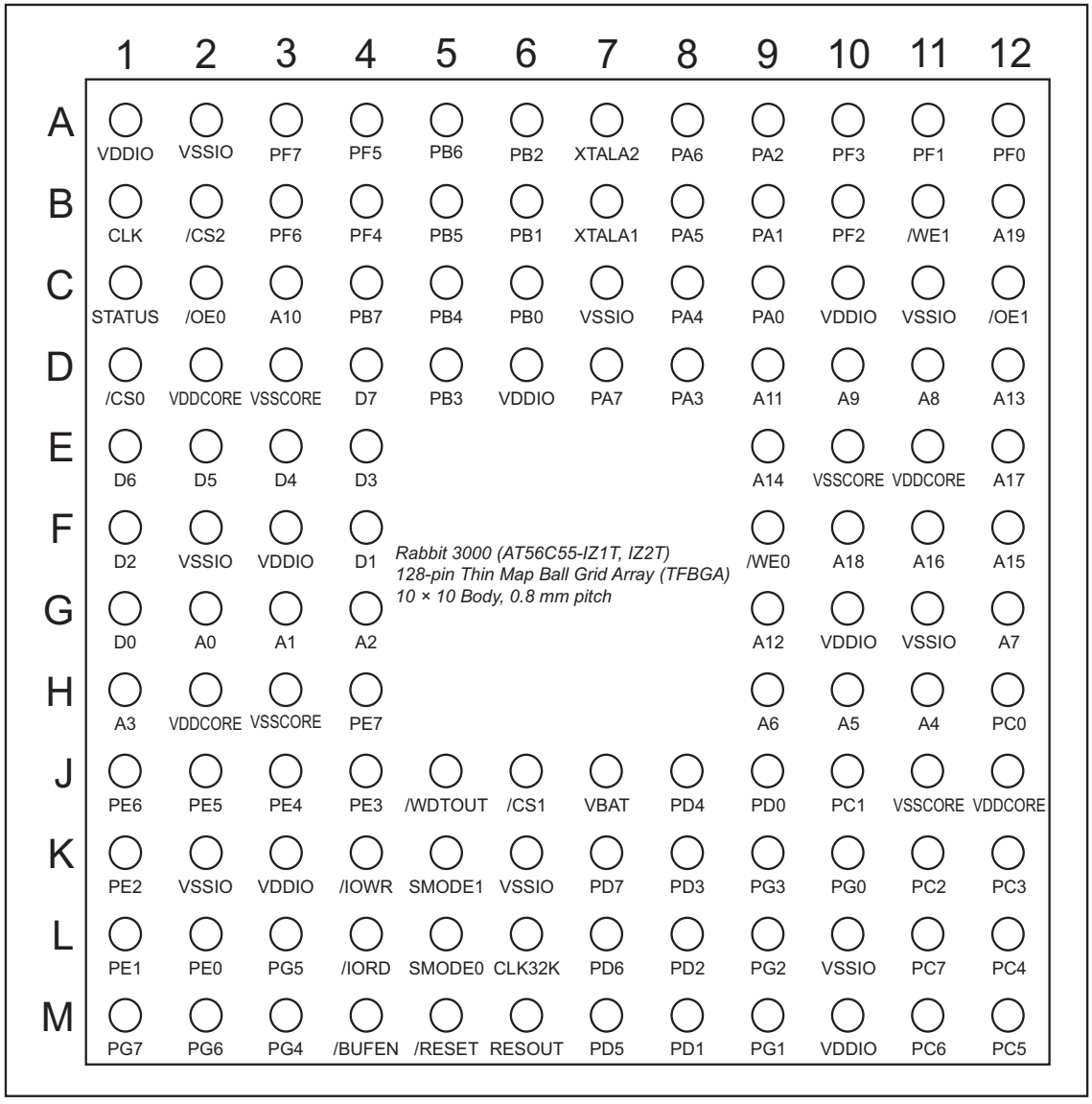


Figure 28-4. Ball Grid Array Pinout Looking Through the Top of Package

28.2.2 Mechanical Dimensions and Land Pattern

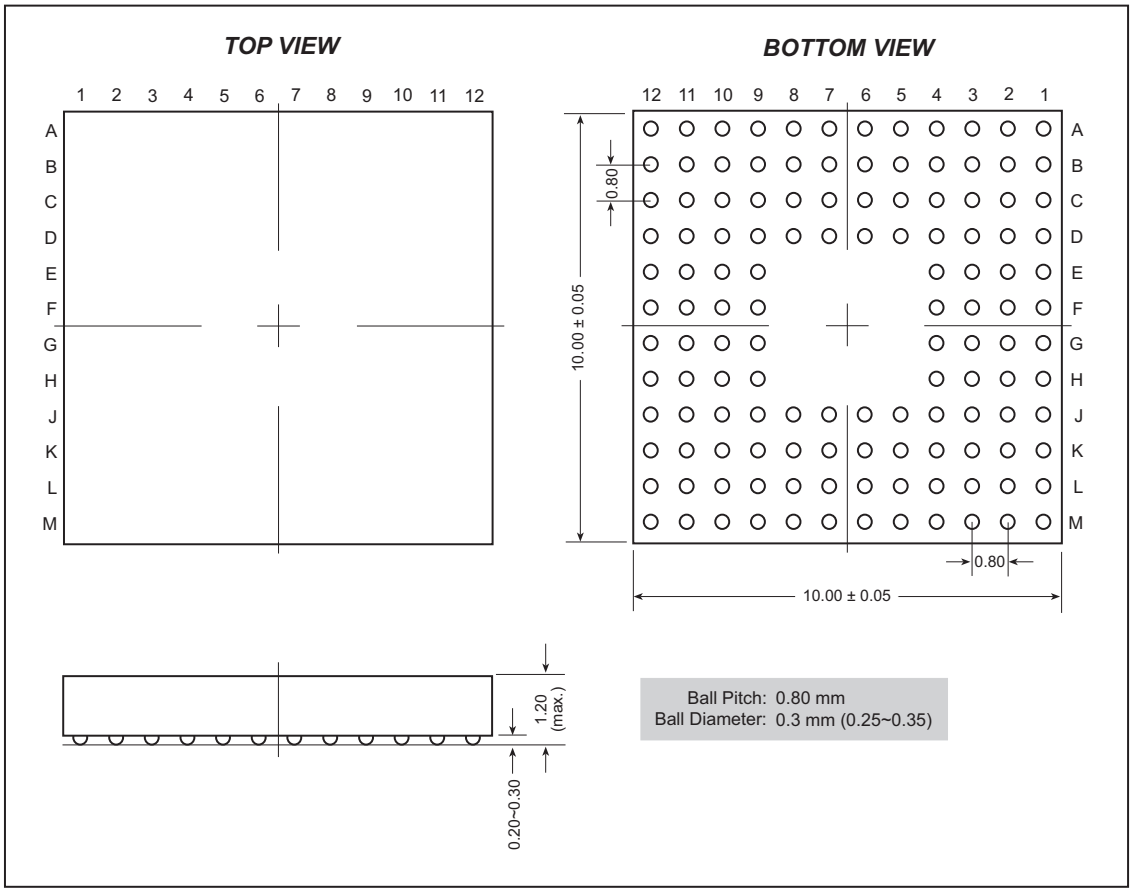


Figure 28-5. BGA Package Outline

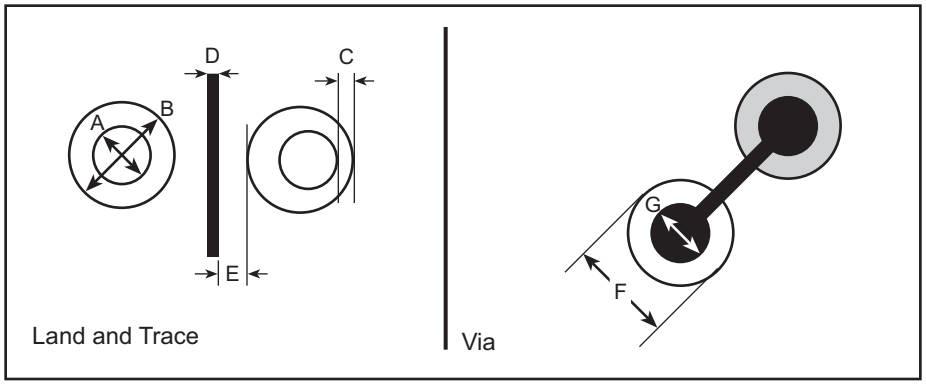
Table 28-1. Ball and Land Size Dimensions

Nominal Ball Diameter (mm)	Tolerance Variation (mm)	Ball Pitch (mm)	Nominal Land Diameter (mm)	Land Variation (mm)
0.3	0.35–0.25	0.8	0.25	0.25–0.20

The design considerations in Table 28-2 are based on 5 mil design rules and assume a single conductor between solder lands.

Table 28-2. Design Considerations (all dimensions in mm)

Key	Feature	Recommendation
A	Solder Land Diameter	0.254 (0.010)
B	NSMD Defined Land Diameter	0.406 (0.016)
C	Land to Mask Clearance (min.)	0.050 (0.002)
D	Conductor Width (max.)	0.127 (0.005)
E	Conductor Spacing (typ.)	0.127 (0.005)
F	Via Capture Pad (max.)	0.406 (0.016)
G	Via Drill Size (max.)	0.254 (0.010)



28.3 Rabbit Pin Descriptions

Table 28-3 lists all the pins on the Rabbit 3000 along with the data direction of the pin, its function, and the pin number on the die.

Table 28-3. Rabbit Pin Descriptions

Pin Group	Pin Name	Direction	Function	Pin Numbers LQFP	Pin Numbers TFBGA
Hardware	CLK	Output	Internal Clock	2	B1
	CLK32K	Input	32 kHz Oscillator In	49	L6
	/RESET	Input	Master Reset	46	M5
	RESOUT	Output	Reset Output	50	M6
	XTALA1	Input	Main Oscillator In—if an external clock is used, this pin should be driven by the external clock; see Technical Note TN235 for more information on external oscillator circuits	113	B7
	XTALA2	Output	Main Oscillator Out	114	A7
CPU Buses	ADDR[19:0]	Output	Address Bus	various	
	DATA[7:0]	Bidirectional	Data Bus	10–15, 18–19	D4, E1–E4, F1, F4, G0
Status/Control	/WDTOUT	Output	WDT Time-Out	43	J5
	STATUS	Output	Instruction Fetch First Byte	4	C1
	SMODE[1:0]	Input	Bootstrap Mode Select	44, 45	K5, L5
Memory Chip Selects	/CS0	Output	Memory Chip Select 0	7	D1
	/CS1	Output	Memory Chip Select 1	47	J6
	/CS2	Output	Memory Chip Select 2	3	B2
Memory Output Enables	/OE0	Output	Memory Output Enable 0	5	C2
	/OE1	Output	Memory Output Enable 1	95	C12
Memory Write Enables	/WE0	Output	Memory Write Enable 0	86	F9
	/WE1	Output	Memory Write Enable 1	99	B11
I/O Control	/BUFEN	Output	I/O Buffer Enable	42	M4
	/IORD	Output	I/O Read Enable	41	L4
	/IOWR	Output	I/O Write Enable	40	K4

Table 28-3. Rabbit Pin Descriptions (continued)

Pin Group	Pin Name	Direction	Function	Pin Numbers LQFP	Pin Numbers TFBGA
I/O ports	PA[7:0]	Input / Output	I/O Port A	111–104	D7, A8, B8, C8, D8, A9, B9, C9
I/O ports (continued)	PB[7:0]	Input / Output	I/O Port B	123–116	C4, A5, B5, C5, D5, A6, B6, C6
	PC[7:0]	4 In / 4 Out	I/O Port C	66–71, 74, 75	L11, M11, M12, L12, K12, K11, J10, H12
	PD[7:0]	Input / Output	I/O Port D	52–59	K7, L7, M7, J8, K8, L8, M8, J9
	PE[7:0]	Input / Output	I/O Port E	26–31, 34, 35	H4, J1–J4, K1, L1–L2
	PF[7:0]	Input / Output	I/O Port F	127–124, 103–100	A3, B3, A4, B4, A10, B10, A11, A12
	PG[7:0]	Input / Output	I/O Port G	36–38, 60–63	M1, M2, L3, M3, K9, L9, M9, K10
Power, processor core	VDDCORE		+3.3 V	8, 24, 72, 88	D2, E11, H2, J12
Power Processor I/O Ring	VDDIO		+3.3 V	1, 17, 33, 65, 81, 97, 115	A1, C10, D6, F3, G10, K3, M10
Power Battery Backup	VBAT		+3.3 V or battery	51	J7
Ground Processor Core	VSSCORE		Ground	9, 25, 73, 89	D3, E10, H3, J11
Ground Processor I/O Ring	VSSIO		Ground	16, 32, 48, 64, 80, 96, 112, 128	A2, C7, C11, F2, G11, K2, K6, L10

APPENDIX A. PARALLEL PORT PINS WITH ALTERNATE FUNCTIONS

A.1 Description of Pins with Alternate Functions

Table A-1. Pins With Alternate Functions

Pin Name	Output Function	Input Function	Input Capture Option
PA[7:0]	SLAVE D[7:0], ID[7:0]	SLAVE D[7:0], ID[7:0]	
PB7	SLAVEATTN, IA5		
PB6	IA4	/ASCS*	
PB5	IA3	SD1	
PB4	IA2	SD0	
PB3	IA1	/SRD	
PB2	IA0	/SWR	
PB1	CLKA	CLKA	
PB0	CLKB	CLKB	
PC7	n/a	RXA	×
PC6	TXA	n/a	
PC5	n/a	RXB	×
PC4	TXB	n/a	
PC3	n/a	RXC	×
PC2	TXC	n/a	
PC1	n/a	RXD	×
PC0	TXD	n/a	
PD7	APWM3*	ARXA	×
PD6	ATXA		
PD5	APWM2*	ARXB	×
PD4	ATXB		
PD3			×
PD2			

Table A-1. Pins With Alternate Functions (continued)

Pin Name	Output Function	Input Function	Input Capture Option
PD1			×
PD0			
PE7	I7	/SCS (slave chip select)	
PE6	I6		
PE5	I5	INT1B	
PE4	I4	INT0B	
PE3	I3		
PE2	I2		
PE1	I1	INT1A	
PE0	I0	INT0A	
PF7	PWM3	AQD2A	×
PF6	PWM2	AQD2B	
PF5	PWM1	AQD1A	×
PF4	PWM0	AQD1B	
PF3		QD2A	×
PF2		QD2B	
PF1	CLKC	QD1A, CLKC	×
PF0	CLKD	QD1B, CLKD	
PG7	APWM1*	RXE	×
PG6	TXE		
PG5	RCLKE	RCLKE, ARXE*	×
PG4	TCLKE	TCLKE, ARCLKE*	
PG3	APWM0*	RXF	×
PG2	TXF		
PG1	RCLKF	RCLKF, ARXF*	×
PG0	TCLKF	TCLKF, ARCLKF*	

* Introduced with Rabbit 3000A chip

APPENDIX B. RABBIT 3000 REVISIONS

Since its release, the Rabbit 3000 microprocessor has gone through one revision. The revision reflects bug fixes, improvements, and the introduction of new features. All Rabbit 3000 revisions are pin-compatible, and transparently replace previous versions of the chip.

The Rabbit 3000 has been supplied in the following versions.

1. **Original Rabbit 3000**—Available in two packages and identified by ***IL1T*** for the LQFP package and ***IZ1T*** for the TFBGA package. The LQFP package began shipping in March 2002, and the TFBGA package began shipping in January 2003. There were several bugs:

    (a) Port A decode bug—This bug is documented in Rabbit's Technical Note TN228, ***Rabbit 3000 Parallel Port F Bug***. The problem involves an incomplete address decode of the data output register for Parallel Port A. If Parallel Port A is used as an output or is used as the bidirectional bus for the slave port, then writing to any of the Parallel Port F registers will cause a spurious write to the Parallel Port A register.

    (b) LDIR/LDDR with wait states—A new **`LDIR/LDDR`** bug was discovered in September, 2002. The problem has to do with wait states and the block move operations. With this problem, the first iteration of **`LDIR/LDDR`** uses the correct number of wait states for both the read and the write. However, all subsequent iterations use the number of waits programmed for the memory located at the write address for both the read and the write cycles. This becomes a problem when moving a block of data from a slow memory device requiring wait states to a fast memory device requiring no wait states. With respect to external I/O operations, the **`LDIR`** or **`LDDR`** performs reads with zero wait states independent of the waits programmed for the I/O for all but the first iteration. The first iteration is correct. This bug is automatically corrected by Dynamic C, and was fixed in future generations of the chip.

    (c) Interrupt after I/O with Short /CSx enabled—When the short chip select option is enabled, the interrupt sequence will attempt to write the return address to the stack if an interrupt takes place immediately after an internal or an external I/O instruction. The chip select will be suppressed during the write cycle, and the correct return address will not be stored on the stack. This happens only when an interrupt takes place immediately after an I/O instruction when the short chip select option is enabled.

(d) IrDA bug—This bug is documented in Rabbit's Technical Note TN236, ***Rabbit 3000 IrDA Bug***. When configured to operate in the IrDA mode, the serial port may at times generate an extra pulse before the start bit is transmitted. This pulse may appear either before a multi-character transmission or before a single-character transmission. If the beginning of the start bit coincides with when the IrDA pulse generator output is high, there will be a spurious 1/16th-bit cell pulse on the transmit output.

2. **First revision (Rabbit 3000A)**—Available in two packages and identified by ***IL2T*** for the LQFP package and ***IZ2T*** for the TFBGA package. This version began shipping in August 2003. All the bugs in the original Rabbit 3000 were fixed. The Rabbit 3000A contains a number of new features and improvements.

   (a) A new mode of operation known as System/User mode was added. This mode provides a framework for separating application code from system-critical code, which helps prevent application code from crashing the entire device. System/User mode is described in detail in Chapter 26.

   (b) The ability to write-protect 64 KB physical memory blocks was added, with the option of further protecting two of the 64 KB blocks in 4 KB segments. Attempts to write to a protected block triggers a Priority 3 write-protection interrupt.

   (c) Stack protection was added. Writing outside set stack boundaries triggers a Priority 3 stack violation interrupt.

   (d) RAM segment relocation was added. This feature allows a 1, 2, or 4 KB segment of the logical memory space to be mapped as data (or for program execution) when separate I/D space is enabled.

   (e) Secondary watchdog timer added. The secondary watchdog timer was added to function as a safety net for the periodic interrupt.

   (f) Two new opcodes were added to support multiply-and-add and multiply-and-subtract operations on large unsigned integers. These operations can be used to speed up public-key calculations.

   (g) Six new opcodes were added to support block-copy operations from I/O addresses to memory addresses and vice-versa.

   (h) The I/O address space has been expanded to 16 bits to make room for new peripherals.

   (i) Two new features were added to further expand the external I/O interface capabilities of the processor. First, an option was added to enable or disable the external I/O bus interface for a given I/O bank. If the external I/O bus is disabled for a given external I/O bank, the processor uses the memory bus for external I/O transactions. The second feature is the addition of an option for enabling hold time for external I/O read operations. The option shortens the read strobes by one clock cycle.

(j) The low-power capability of the processor was further expanded with the addition of short chip select timing for all clock modes (except for divide-by-one mode) and for reads, writes, or both.

(k) The PWM outputs can now trigger a PWM interrupt each cycle or every other/fourth/eighth cycle. In addition, the PWM output can be suppressed every other cycle, three out of every four cycles, or seven out of every eight cycles. These options were added to provide support for driving servos in addition to generating audio using the Rabbit 3000A.

(l) The quadrature decoder hardware can be configured to use a 10-bit counter in place of the existing 8-bit counter.

(m) An option was added to alternatively multiplex PWM outputs, slave chip select (/SCS), and Serial Ports E and F transmit and receive clocks on other pins.

(n) The Schmitt trigger IC normally required for the low power 32.768 kHz oscillator circuit is now integrated inside the Rabbit 3000A.

NOTE: Based on this modification, a new low-power oscillator circuit is recommended for use with Rabbit 3000A-based systems. Please refer to TN235, ***External 32.768 kHz Oscillator Circuits***, for more information on the circuit.

Rabbit 3000 chips identified by ***UL2T*** for the LQFP package and ***DZ2J*** for the TFBGA package are RoHS-compliant. The ***UL2T*** and ***DZ2J*** RoHS versions were introduced in mid-2007, and the ***IL2T*** and ***IZ2T*** non-RoHS versions are no longer avaialble.

B.1 Discussion of Fixes and Improvements

Table B-1 lists the bug fixes, improvements, and additions for the various revisions of the Rabbit 3000.

Table B-1. Summary of Rabbit 3000 Improvements and Fixes

Description	Rabbit 3000 (IL1T/IZ1T)	Rabbit 3000A (IL2T/IZ2T)
ID Registers for version/revision identification.	X	X
System/User mode.		X
Memory protection scheme.		X
Stack protection.		X
RAM segment relocation.		X
Secondary watchdog timer.		X
Multiply-add and multiply-subtract.		X
Variants of block move opcodes.		X
16-bit internal I/O address space.		X
External I/O interface enhancements.		X
Expanded low-power capability.		X
PWM improvements.		X
Quadrature decoder improvements.		X
Integrated Schmitt trigger for 32 kHz oscillator input.		X
Alternate output port connection for numerous peripherals.		X
Port A decode bug fix.		X
LDIR/LDDR with wait states bug fix.		X
Interrupt after I/O with short /CSx enabled bug fix.		X
IrDA bug fix.		X

B.1.1 Rabbit Internal I/O Registers

Table B-2 summarizes the reset state of the new I/O registers added in the Rabbit 3000A revision. Table B-2 summarizes the reset state of the existing I/O registers with new features.

Table B-2. Reset State of New Rabbit 3000A I/O Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Secondary Watchdog Timer Register	SWDTR	0x000C	W	11111111
RAM Segment Register	RAMSR	0x0448	W	00000000
Write Protect Control Register	WPCR	0x0440	W	00000000
Stack Limit Control Register	STKCR	0x0444	W	00000000
Stack Low Limit Register	STKLLR	0x0445	W	xxxxxxxx
Stack High Limit Register	STKHLR	0x0446	W	xxxxxxxx
Write Protect Low Register	WPLR	0x0460	W	00000000
Write Protect High Register	WPHR	0x0461	W	00000000
Write Protect Segment A Register	WPSAR	0x0480	W	00000000
Write Protect Segment A Low Register	WPSALR	0x0481	W	00000000
Write Protect Segment A High Register	WPSAHR	0x0482	W	00000000
Write Protect Segment B Register	WPSBR	0x0484	W	00000000
Write Protect Segment B Low Register	WPSBLR	0x0485	W	00000000
Write Protect Segment B High Register	WPSBHR	0x0486	W	00000000
Real Time Clock User Enable Register	RTUER	0x0300	W	00000000
Slave Port User Enable Register	SPUER	0x0320	W	00000000
Parallel Port A User Enable Register	PAUER	0x0330	W	00000000
Parallel Port B User Enable Register	PBUER	0x0340	W	00000000
Parallel Port C User Enable Register	PCUER	0x0350	W	00000000
Parallel Port D User Enable Register	PDUER	0x0360	W	00000000
Parallel Port E User Enable Register	PEUER	0x0370	W	00000000
Parallel Port F User Enable Register	PFUER	0x0338	W	00000000
Parallel Port G User Enable Register	PGUER	0x0348	W	00000000
Input Capture User Enable Register	ICUER	0x0358	W	00000000
I/O Bank User Enable Register	IBUER	0x0380	W	00000000
PWM User Enable Register	PWUER	0x0388	W	00000000
Quad Decode User Enable Register	QDUER	0x0390	W	00000000

Table B-2. Reset State of New Rabbit 3000A I/O Registers (continued)

Register Name	Mnemonic	I/O Address	R/W	Reset
External Interrupt User Enable Register	IUER	0x0398	W	00000000
Timer A User Enable Register	TAUER	0x03A0	W	00000000
Timer B User Enable Register	TBUER	0x03B0	W	00000000
Serial Port A User Enable Register	SAUER	0x03C0	W	00000000
Serial Port B User Enable Register	SBUER	0x03D0	W	00000000
Serial Port C User Enable Register	SCUER	0x03E0	W	00000000
Serial Port D User Enable Register	SDUER	0x03F0	W	00000000
Serial Port E User Enable Register	SEUER	0x03C8	W	00000000
Serial Port F User Enable Register	SFUER	0x03D8	W	00000000
Enable Dual-Mode Register	EDMR	0x0420	W	00000000
Quad Decode Count1 High Register	QDC1HR	0x0095	R	xxxxxxxx
Quad Decode Count 2 High Register	QDC2HR	0x0097	R	xxxxxxxx

Table B-3. Reset State of I/O Registers Modified in Rabbit 3000A

Register Name	Mnemonic	I/O Address	R/W	Rabbit 3000 Reset	Rabbit 3000A Reset
Global Power Save Control Register	GPSCR	0x000D	W	0000x000	00000000
Global Revision Register	GREV	0x002F	R	0xx00000	0xx00001
MMU Expanded Code Register	MECR	0x0018	R/W	xxxxx000	00000000
Memory Timing Control Register	MTCR	0x0019	W	xxxx0000	00000000
Breakpoint/Debug Control Register	BDCR	0x001C	W	0xxxxxxx	00000000
I/O Bank 0 Control Register	IB0CR	0x0080	W	000000xx	00000000
I/O Bank 1 Control Register	IB1CR	0x0081	W	000000xx	00000000
I/O Bank 2 Control Register	IB2CR	0x0082	W	000000xx	00000000
I/O Bank 3 Control Register	IB3CR	0x0083	W	000000xx	00000000
I/O Bank 4 Control Register	IB4CR	0x0084	W	000000xx	00000000
I/O Bank 5 Control Register	IB5CR	0x0085	W	000000xx	00000000
I/O Bank 6 Control Register	IB6CR	0x0086	W	000000xx	00000000
I/O Bank 7 Control Register	IB7CR	0x0087	W	000000xx	00000000
PWM LSB 0 Register	PWL0R	0x0088	W	xxxxxxxx	xxxxx00x
PWM LSB 1 Register	PWL1R	0x008A	W	xxxxxxxx	xxxxx00x
PWM LSB 2 Register	PWL2R	0x008C	W	xxxxxxxx	xxxxx00x
PWM LSB 3 Register	PWL3R	0x008E	W	xxxxxxxx	xxxxx00x
Quad Decode Control Register	QDCR	0x0091	W	00xx0000	00000000

B.1.2 Peripheral and ISR Address

Table B-4. Rabbit 3000 I/O Address Ranges and Interrupt Service Vectors

On-Chip Peripheral	I/O Address Range	ISR Starting Address
System Management	0x0000–0x000F	{R[7:1], 0, 0x0000}
Memory Management	0x0010–0x001F and 0x0400–0x04FF	No interrupts
Slave Port	0x0020–0x002F	{IIR[7:1], 0, 0x0080}
Parallel Port A	0x0030–0x0037	No interrupts
Parallel Port F	0x0038–0x003F	No interrupts
Parallel Port B	0x0040–0x0047	No interrupts
Parallel Port G	0x0048–0x004F	No interrupts
Parallel Port C	0x0050–0x0055	No interrupts
Input Capture	0x0056–0x005F	{IIR[7:1], 1, 0x00A0}
Parallel Port D	0x0060–0x006F	No interrupts
Parallel Port E	0x0070–0x007F	No interrupts
External I/O Control	0x0080–0x0087	No interrupts
Pulse Width Modulator	0x0088–0x008F	{IIR[7:1], 1, 0x0070}
Quadrature Decoder	0x0090–0x0097	{IIR[7:1], 1, 0x0090}
External Interrupts	0x0098–0x009F	INT0 {EIR, 0x0000} INT1 {EIR, 0x0010}
Timer A	0x00A0–0x00AF	{IIR[7:1], 0, 0x00A0}
Timer B	0x00B0–0x00BF	{IIR[7:1], 0, 0x00B0}
Serial Port A (async/cks)	0x00C0–0x00C7	{IIR[7:1], 0, 0x00C0}
Serial Port E (async/HDLC)	0x00C8–0x00CF	{IIR[7:1], 1, 0x00C0}
Serial Port B (async/cks)	0x00D0–0x00D7	{IIR[7:1], 0, 0x00D0}
Serial Port F (async/HDLC)	0x00D8–0x00DF	{IIR[7:1], 1, 0x00D0}
Serial Port C (async/cks)	0x00E0–0x00E7	{IIR[7:1], 0, 0x00E0}
Serial Port D (async/cks)	0x00F0–0x00F7	{IIR[7:1], 0, 0x00F0}
RST 10 instruction	n/a	{IIR[7:1], 0, 0x0020}
RST 18 instruction	n/a	{IIR[7:1], 0, 0x0030}
RST 20 instruction	n/a	{IIR[7:1], 0, 0x0040}
RST 28 instruction	n/a	{IIR[7:1], 0, 0x0050}

Table B-4. Rabbit 3000 I/O Address Ranges and Interrupt Service Vectors (continued)

On-Chip Peripheral	I/O Address Range	ISR Starting Address
SYSCALL instruction	n/a	{IIR[7:1], 0, 0x0060}
RST 38 instruction	n/a	{IIR[7:1], 0, 0x0070}
Secondary Watchdog	0x000C	{IIR[7:1], 0, 0x0010}
Stack Limit Violation	n/a	{IIR[7:1], 1, 0x00B0}
Write Protection Violation	n/a	{IIR[7:1], 0, 0x0090}
System Mode Violation	n/a	{IIR[7:1], 1, 0x0080}

B.1.3 Revision-Level ID Register

Two read-only registers are provided to allow software to identify the Rabbit microprocessor and recognize the features and capabilities of the chip. Five bits in each of these registers are unique to each version of the chip. One register identifies the CPU (GCPU), and the other register is reserved for revision identification (GREV). The CPU identification (GCPU) of all revisions of the Rabbit 3000 microprocessor is the same. Rabbit 3000 revisions are differentiated by the value in the GREV register.

Table B-5 summarizes the processor identification information for the different Rabbit 3000 versions.

Table B-5. Rabbit 3000 Revision Identification Information

Processor Revision	Package Identifier	GCPU [4:0]	GREV [4:0]
Rabbit 3000	IL1T, IZ1T	00001	00000
Rabbit 3000A	IL2T, IZ2T	00001	00001

B.1.4 System/User Mode

By default, all of the hardware is accessible by the programmer. However, if a control bit in the Enable Dual Mode Register (EDMR) is set to one, two operating modes, System and User, become available. The System mode is just like the normal operating mode, but the User mode restricts program access to the hardware and to the System mode. Individual peripherals may be enabled for User mode access in the User Enable registers listed below. When enabled for User mode access, a peripheral interrupt (if it is capable of generating an interrupt) can only be requested at interrupt priority level -2 or -1, and it is assumed that the interrupt service routine will be executed by User mode code. Note that the processor automatically enters the System mode when entering the ISR area in response to an interrupt, and the User mode must be specifically entered before continuing with the interrupt service routine. The System/User mode is discussed in great detail in Chapter 26.

B.1.5 Memory Protection

The ability to inhibit writes to physical memory was added. The sixteen 64 KB physical memory blocks can be individually protected, and two of those blocks can additionally be subdivided and protected at a granularity of 4 KB. When a write is attempted, a new Priority 3 write-protection interrupt request is generated.

The write-protection can be enabled for the User mode only or for all modes (see Chapter 26 for more information).

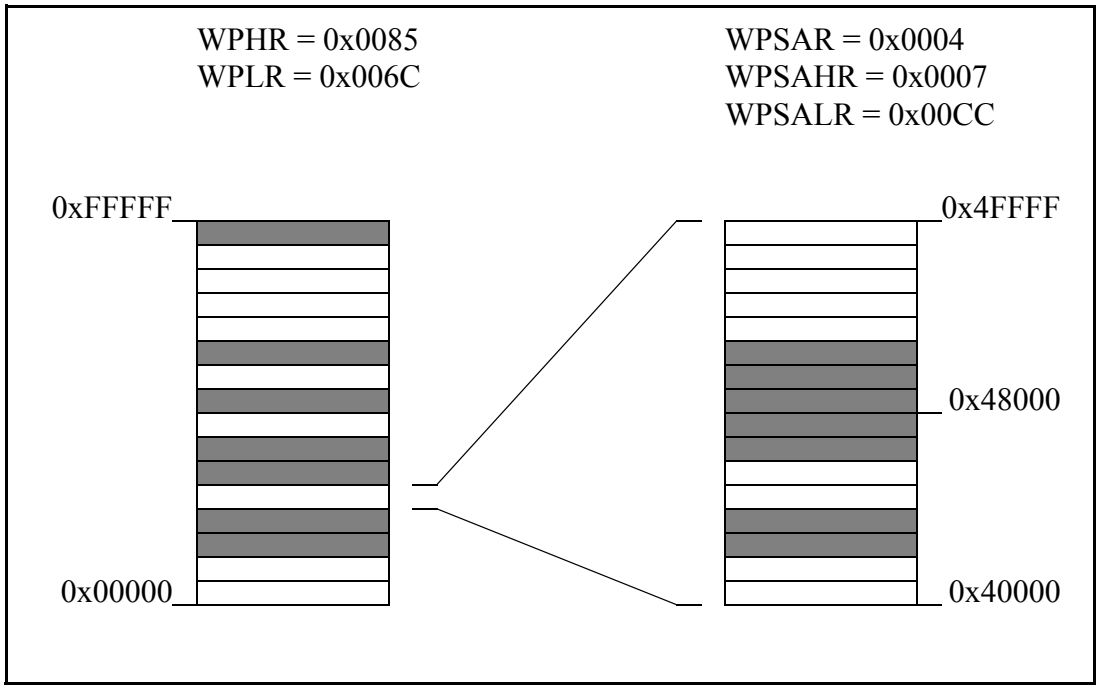


Figure B-1. Sample Memory Protection Layout

B.1.6 Stack Protection

Stack overflow and underflow can now be detected. Low and high stack limits can be set on 256-byte boundaries. When a stack-relative memory access occurs within 16 bytes of these limits (or outside of them), a new Priority 3 stack violation interrupt occurs. The 16-byte buffer exists to allow stack protection even if the stack is placed against a memory segment boundary.

Figure B-2 shows one possible stack layout. A 2048-byte stack is set up by setting STKHLR to 0x00E0, STKLLR to 0x00D8, and SP to 0xDFF0. Any stack-relative memory accesses above 0xDFEF (i.e., stack underflow) or below 0xD810 (i.e., overflow) would trigger the stack violation interrupt.

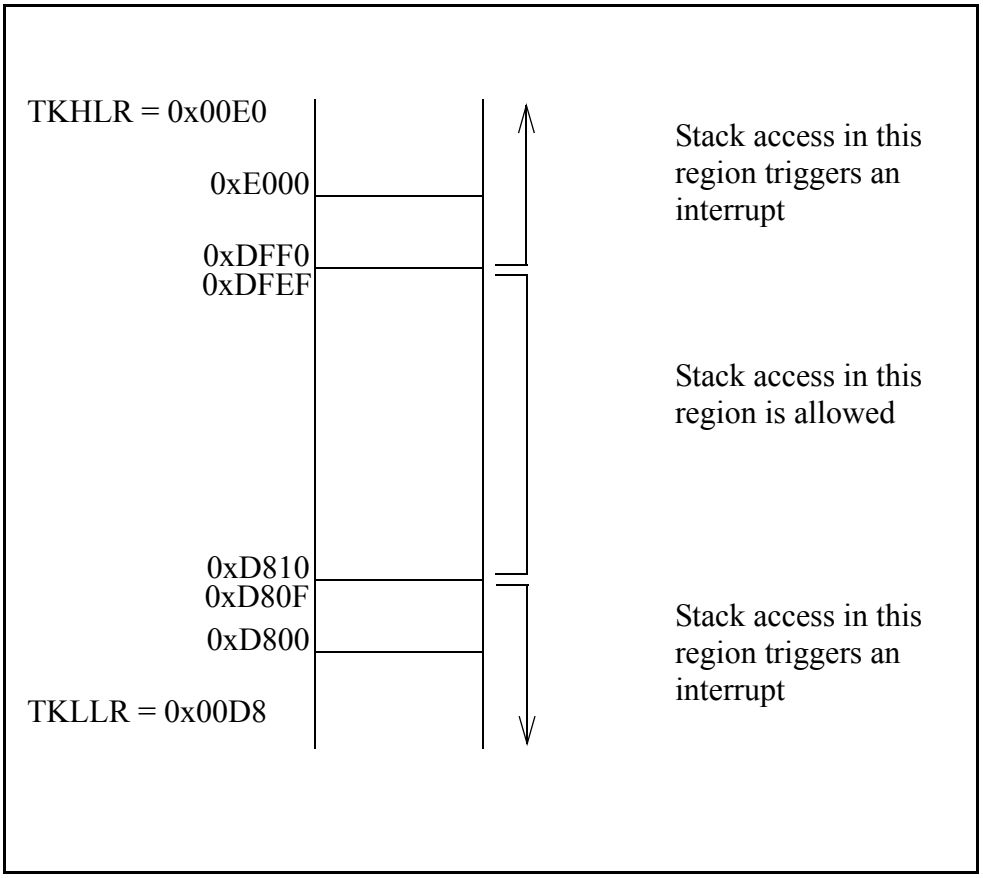


Figure B-2. Simple Stack Protection Layout

B.1.7 RAM Segment Relocation

Normally when instruction/data separation is enabled, instructions are stored in flash memory and data are stored in RAM memory. This can present a problem for the Interrupt Service Routine area, which often requires run-time modification. The RAM Segment Register (RAMSR) allows a 1, 2, or 4 KB segment of the logical memory space to be mapped as data would be mapped, even for program execution.

B.1.8 Secondary Watchdog Timer

The secondary watchdog timer (SWDT) is an eight-bit modulo n + 1 counter clocked by the 32.768 kHz clock. The timer is off by default, and is enabled by writing a 0x005F to the WDTCR. The secondary watchdog timer register (SWDTR) holds the time constant value. Depending on the value loaded into the SWDTR, the timer can request an interrupt anywhere from 30.5 µs to 7.8 ms. If a 0x005F is written to the WDTCR prior to end of the countdown period, the timer will not request an interrupt. If the counter counts down to zero, a level-3 interrupt is generated. The SWDT is intended as a safety net for the periodic interrupt, and would normally be restarted in the service routine for the periodic interrupt. Although the hardware was intended to primarily be used by an operating system when the System/User mode is enabled, it can be used as a configurable periodic interrupt as well.

B.1.9 New Opcodes

Eight new opcodes were added to the Rabbit 3000A. UMA and UMS allow multiply-and-add and multiply-and-subtract operations on large integers, and were added to speed up common cryptographic math used in public-key calculations. The remaining six expand the block copy operations available, especially to and from I/O addresses (internal and external). These opcodes are listed in Table B-6.

Table B-6. New Rabbit 3000 Opcodes

Instruction	Bytes	clk	A	I	S	Z	V	C	Operation
UMA	2	8+8i		-	-	-	-	*	{CY:DE':(HL) = (IX) + [(IY) * DE + DE' + CY]; BC = BC-1; IX = IX+1; IY = IY+1; HL = HL+1; repeat while BC !=0
UMS	2	8+8i		-	-	-	-	*	{CY:DE:(HL) = (IX) - [(IY) * DE + DE' + CY]; BC = BC-1; IX = IX+1; IY = IY+1; HL = HL+1; repeat while BC !=0
LDDSR	2	6+7i		d	-	-	*	-	(DE) = (HL); BC = BC - 1; HL = HL - 1; repeat while BC != 0
LDISR	2	6+7i		d	-	-	*	-	(DE) = (HL); BC = BC - 1; HL = HL + 1; repeat while BC != 0
LSDR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE - 1; HL = HL - 1; repeat while BC != 0
LSIR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE + 1; HL = HL + 1; repeat while BC != 0
LSDDR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE - 1; repeat while BC != 0
LSIDR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE + 1; repeat while BC != 0

B.1.9.1 New UMA/UMS Opcodes

The new **UMA** and **UMS** opcodes perform the following operation:

```
{CY:DE':(HL)} = (IX) ± [(IY) * DE + DE' + CY];
```

where HL, IX, and IY increment after each byte, repeated BC times. This fundamental operation allows the addition or subtraction of two arbitrarily-long unsigned integers after one is scaled by a single-byte value. This operation is common in many cryptographic operations.

B.1.9.2 New Block Copy Opcodes

The LDxR family of block move opcodes has been expanded. In the Rabbit 3000 processor, block copy operations could only be done between memory addresses, or from memory to an I/O address. In addition, the destination I/O address would increment (or decrement if using LDDR) after each byte, making the block copy opcodes effectively useless for repeated reads or writes to a peripheral (for example, a device on the external I/P bus).

Six new block copy opcodes were added to the Rabbit 3000 revision. These opcodes can copy from an I/O address as well as to one, and either the source or destination address can remain fixed instead of changing after each byte. The new opcodes are described in Table B-7.

Table B-7. Rabbit 3000 Revision Block Copy Opcode Effects

Opcode	Source Address Change	Destination Address Change	IOI/IOE Affects
LDDR	-	-	destination
LDIR	+	+	destination
LDDSR	-	none	destination
LDISR	+	none	destination
LSDR	-	-	source
LSIR	+	+	source
LSDDR	none	-	source
LSIDR	none	+	source

B.1.10 Expanded I/O Memory Addressing

In the Rabbit 3000, only the lower 8 bits of an I/O address were decoded. To provide room for new peripherals, this was expanded to 16 bits. To ensure backwards compatibility, the processor always comes up in 8-bit I/O address mode; the 16-bit I/O address mode needs to be enabled in the MMIDR register by setting bit 7 to 1. Bit 7 was always written with a zero in the original Rabbit 3000 chip.

B.1.11 External I/O Improvements

Three new features have been added to the external I/O strobes: the ability to invert the strobe signal, the ability to shorten a read strobe by one clock, and the ability to direct a strobe to either the alternate I/O bus (if enabled) or the memory bus.

B.1.12 Short Chip Select Timing for Writes

The Rabbit 3000 provided the ability to produce shorter chip select strobes for reads when in a reduced-speed mode. A new feature has been added via bits 1:0 to produce short chip select strobes for writes as well, and can be controlled by the GPCSR register.

B.1.12.1 Clock Select and Power Save Modes

Table B-9 outlines the power save modes available in the Rabbit 3000A. The GCSR is shown in Table B-8 for reference.

Table B-8. Global Control/Status Register

Global Control/Status Register (GCSR) (Address = 0x0000)		
Bit(s)	**Value**	**Description**
7:6 (Read-only)	00	No reset or watchdog timer time-out since the last read.
	01	The watchdog timer timed out. These bits are cleared by a read of this register.
	10	This bit combination is not possible.
	11	Reset occurred. These bits are cleared by a read of this register.
5	0	No effect on the periodic interrupt. This bit will always be read as zero.
	1	Force a periodic interrupt to be pending.
4:2	xxx	See table below for decode of this field.
1:0	00	Periodic interrupts are disabled.
	01	Periodic interrupts use Interrupt Priority 1.
	10	Periodic interrupts use Interrupt Priority 2.
	11	Periodic interrupts use Interrupt Priority 3.

Table B-9. Clock Select Field of GCSR

Clock Select Bits 4:2 GCSR	CPU Clock	Peripheral Clock	Main Oscillator	Power-Save CS if Enabled by GPSCR
000	osc/8	osc/8	on	short CS option
001	osc/8	osc	on	short CS option
010	osc	osc	on	none
011	osc/2	osc/2	on	short CS option
100	32 kHz or fraction	32 kHz or fraction	on	self-timed option short CS option
101	32 kHz or fraction	32KHz or fraction	off	self-timed option short CS option
110	osc/4	osc/4	on	short CS option
111	osc/6	osc/6	on	short CS option

B.1.12.2 Short Chip Select Timing

When short chip selects are enabled for read cycles, the chip select signals are active only for the last part of the bus cycle. Wait states are inserted between T1 and T2, so this will have no effect on the duration of the chip select signals in this mode. The timing diagrams below illustrate the actual timing for the different divided cases. In these cases the chip selects are two clock cycles (of the fast oscillator) long.

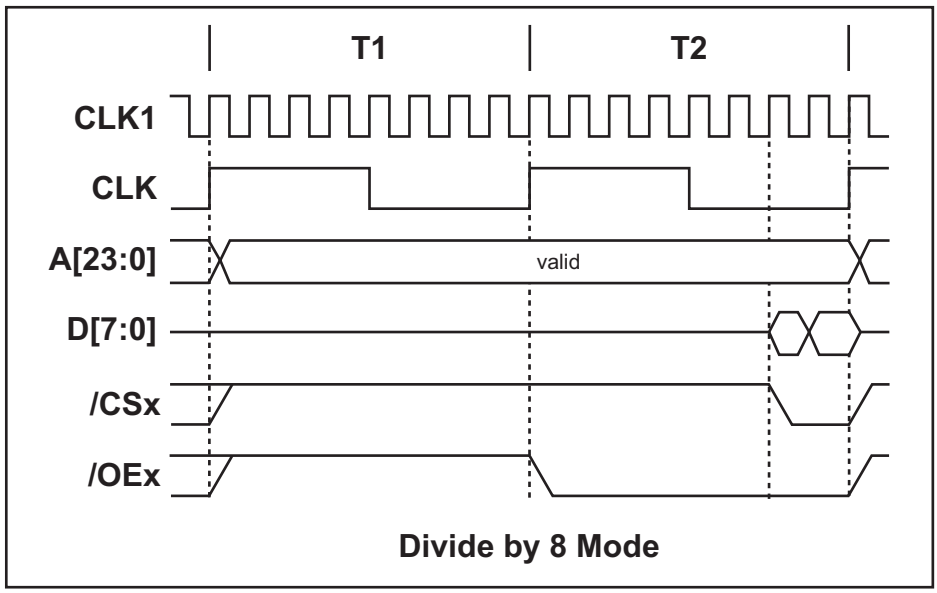


Figure B-3. Short Chip Select Timing: perclk/8, Read Operation

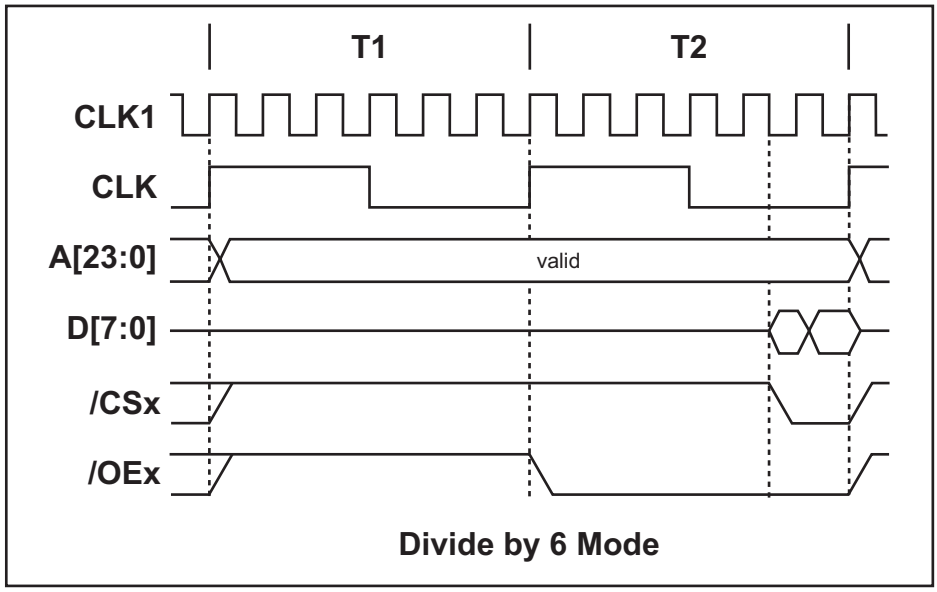


Figure B-4. Short Chip Select Timing: perclk/6, Read Operation

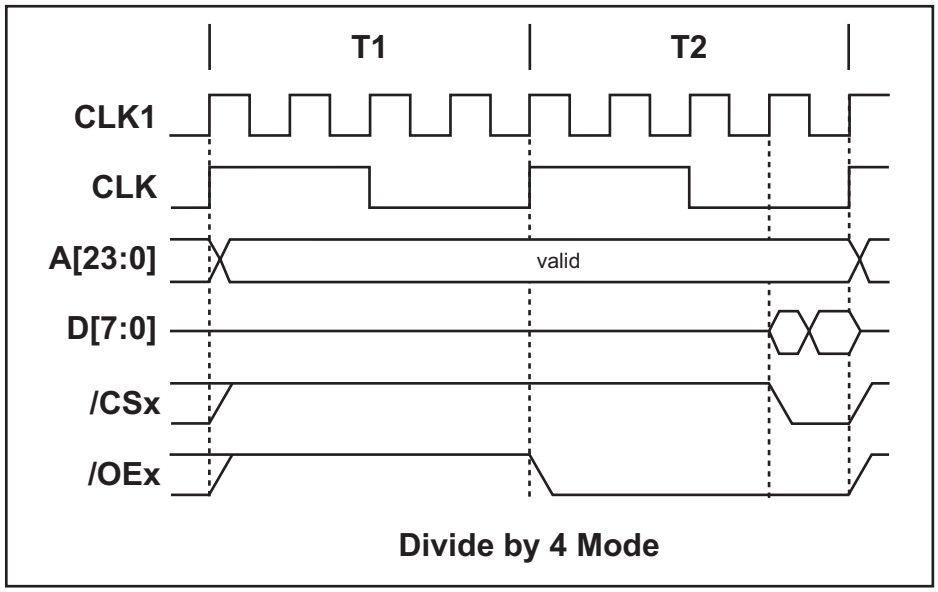


Figure B-5. Short Chip Select Timing: perclk/4, Read Operation

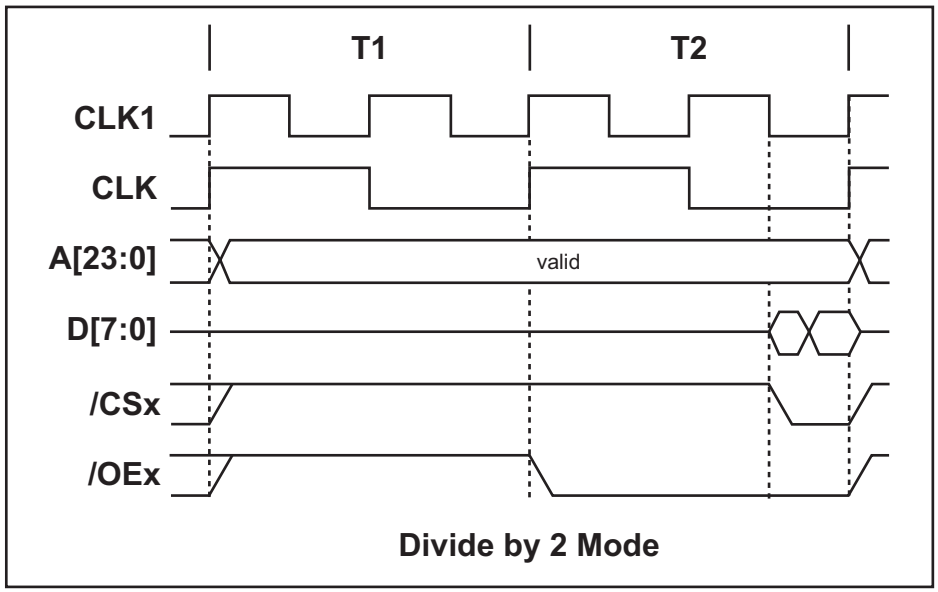


Figure B-6. Short Chip Select Timing: perclk/2, Read Operation

When operating from the 32 kHz oscillator, the same options are available, but the timing is somewhat different. This is illustrated in the diagrams below for the four different cases. In these case the chip selects are one clock cycle (of the 32 kHz clock) long.

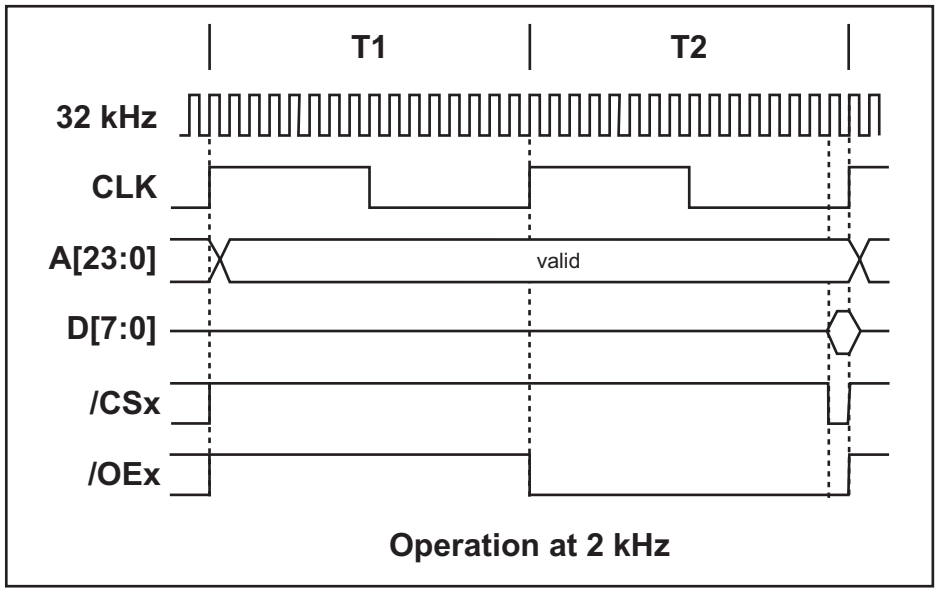


Figure B-7. Short Chip Select Timing: 2 kHz, Read Operation

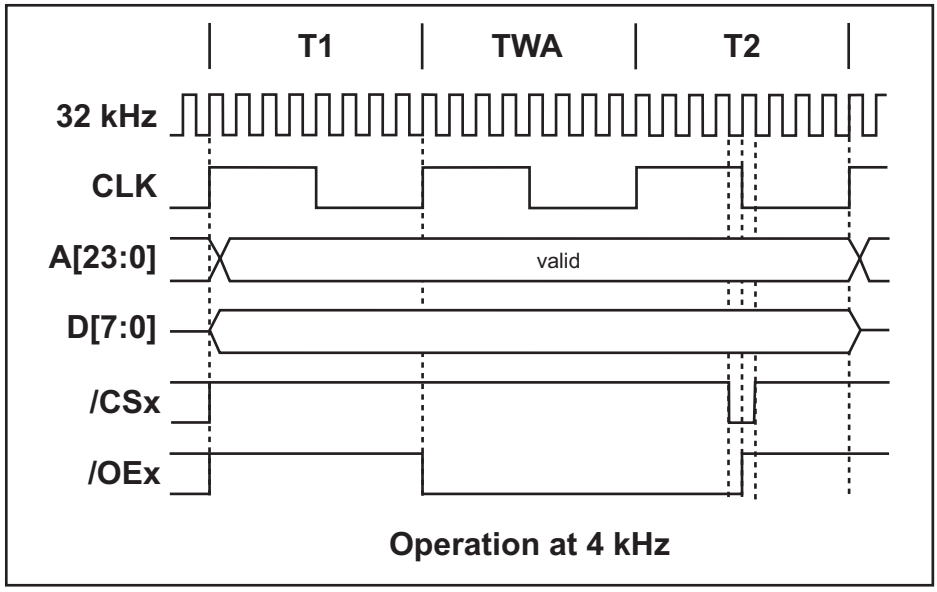


Figure B-8. Short Chip Select Timing: 4 kHz, Read Operation

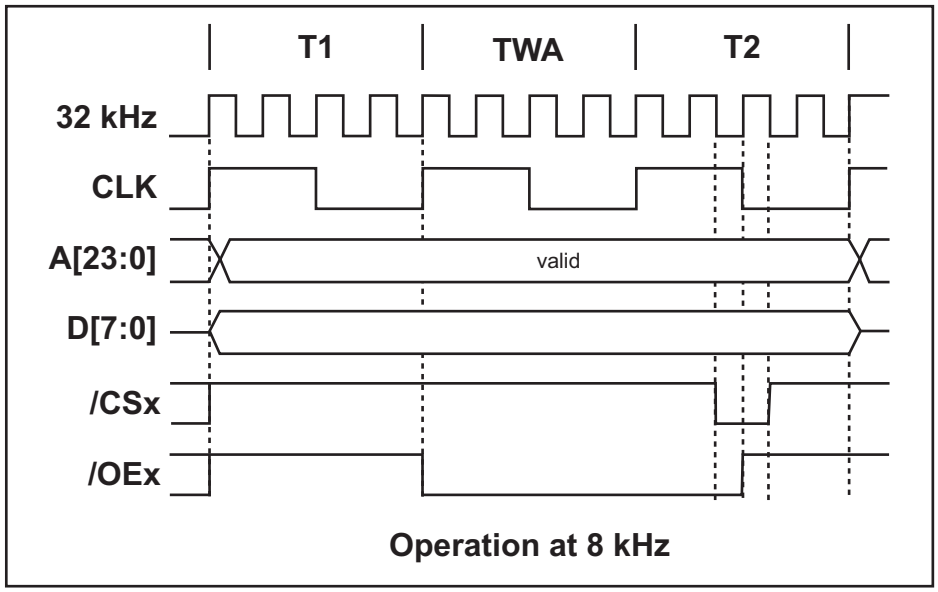


Figure B-9. Short Chip Select Timing: 8 kHz, Read Operation

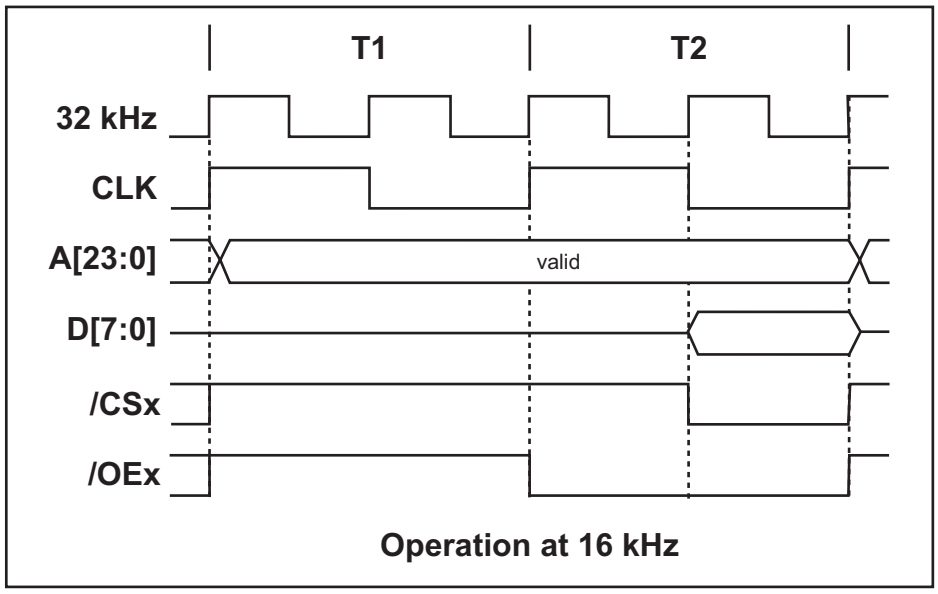


Figure B-10. Short Chip Select Timing: 16 kHz, Read Operation

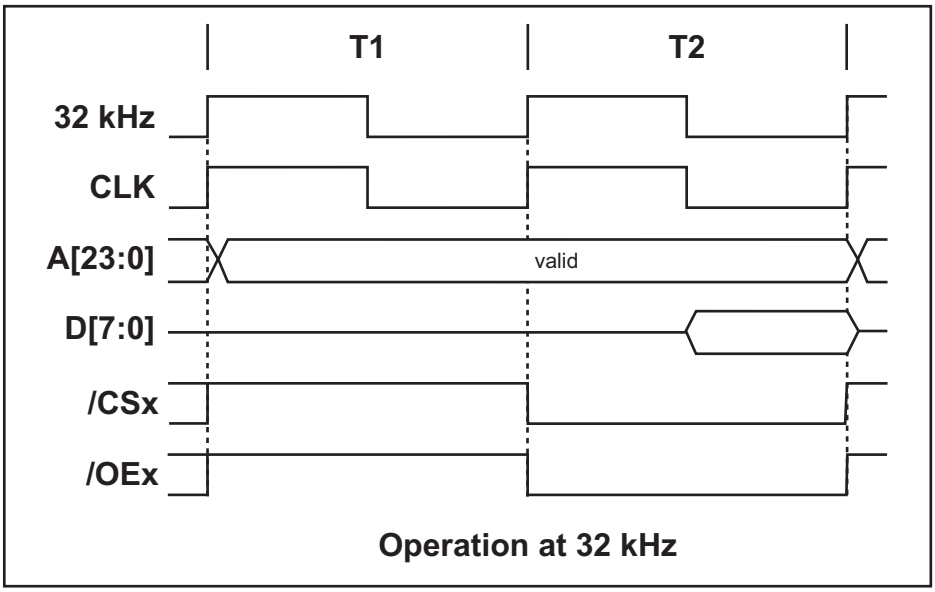


Figure B-11. Short Chip Select Timing: 32 kHz, Read Operation

In the case of write cycles, the chip select signals are active only around the trailing edge of the write signal. Wait states are inserted between T1 and T2, and this will have no effect on the duration of the chip select signals in this mode. The timing diagrams below illustrate the actual timing for the different divided cases. In these cases the chip selects are active for two clock cycles before and two clock cycles after the trailing edge of the write signal.

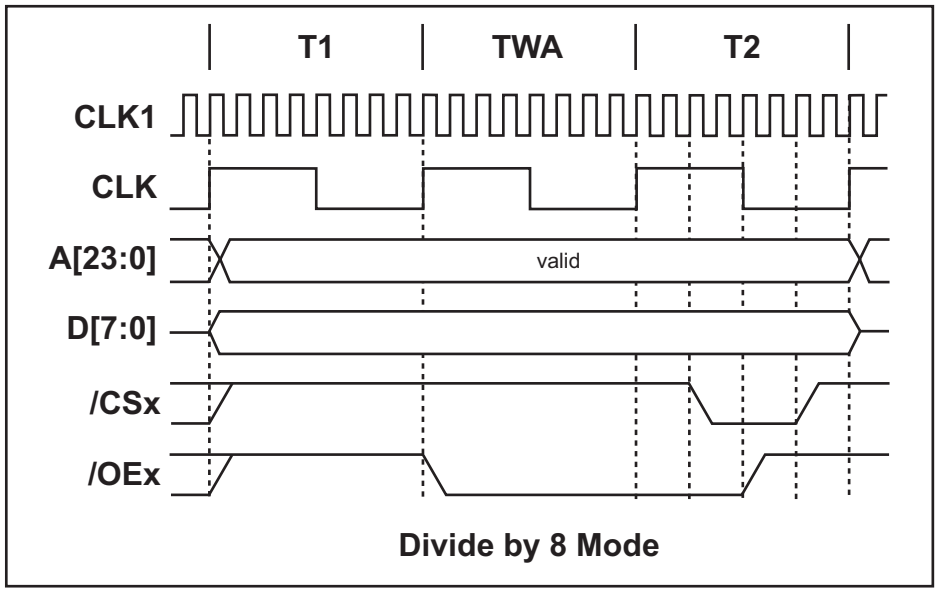


Figure B-12. Short Chip Select Timing: perclk/8, Write Operation

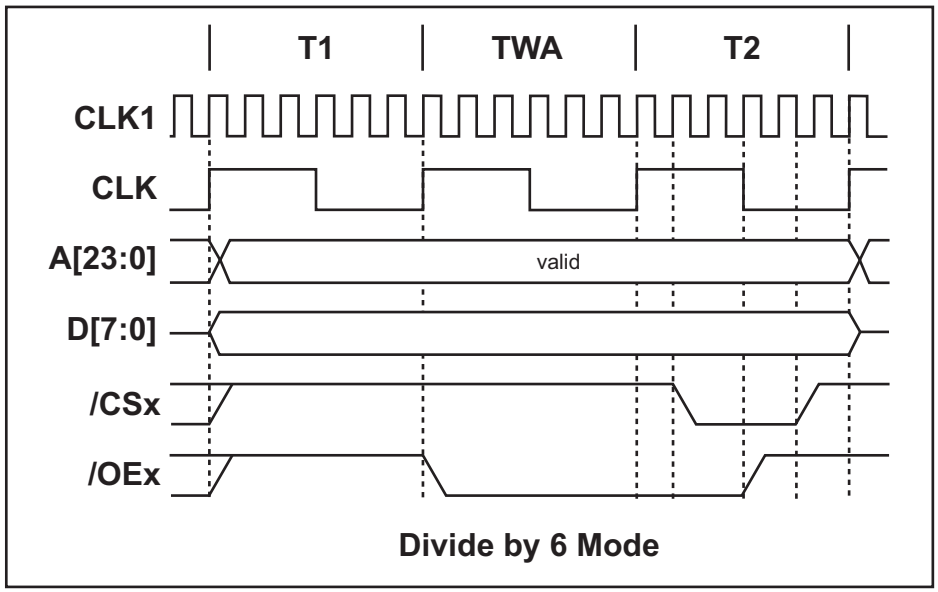


Figure B-13. Short Chip Select Timing: perclk/6, Write Operation

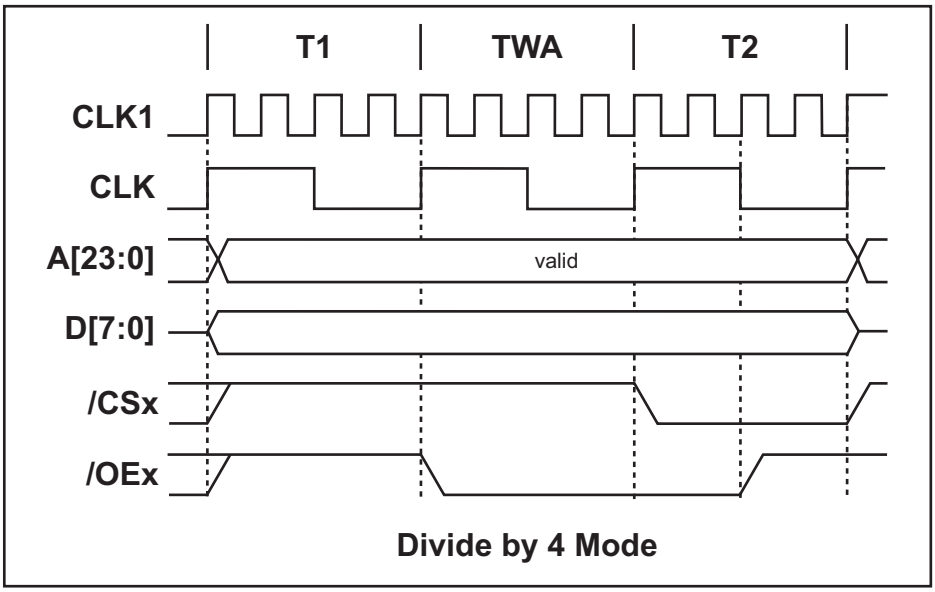


Figure B-14. Short Chip Select Timing: perclk/4, Write Operation

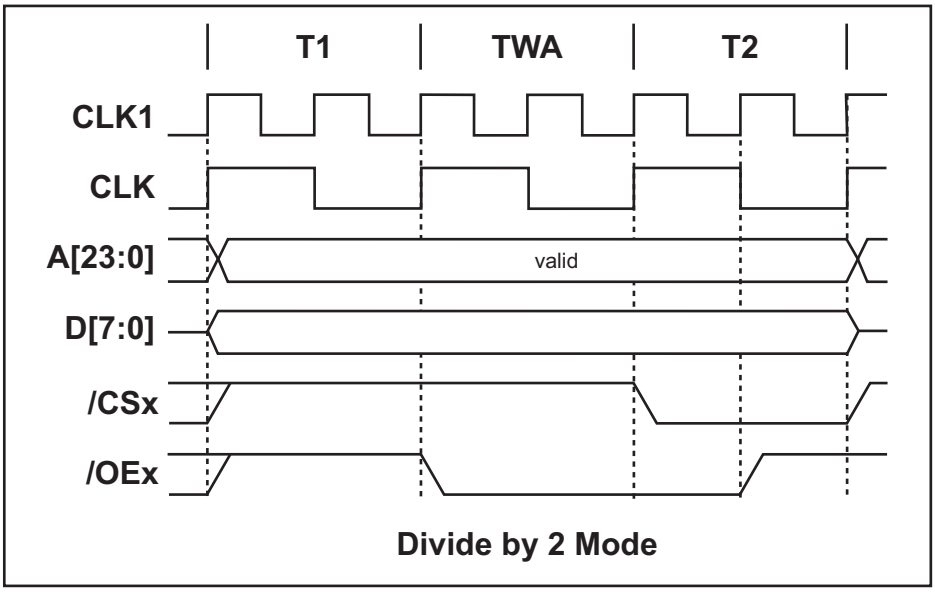


Figure B-15. Short Chip Select Timing: perclk/2, Write Operation

The timing diagrams below illustrate the actual timing for the 32KHz cases of write cycles. In these cases the chip selects are active for one clock cycle before and one clock cycle after the trailing edge of the write signal.

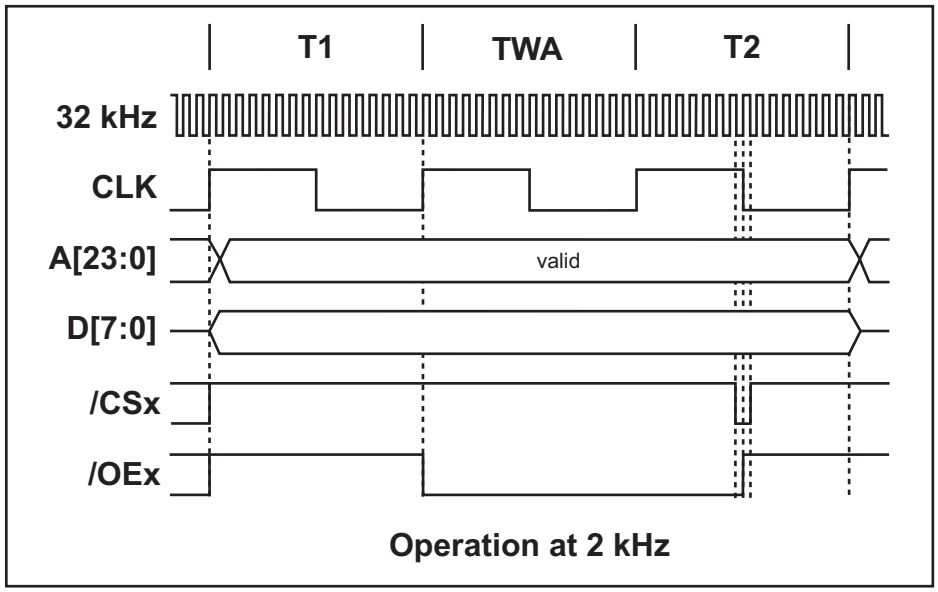


Figure B-16. Short Chip Select Timing: 2 kHz, Write Operation

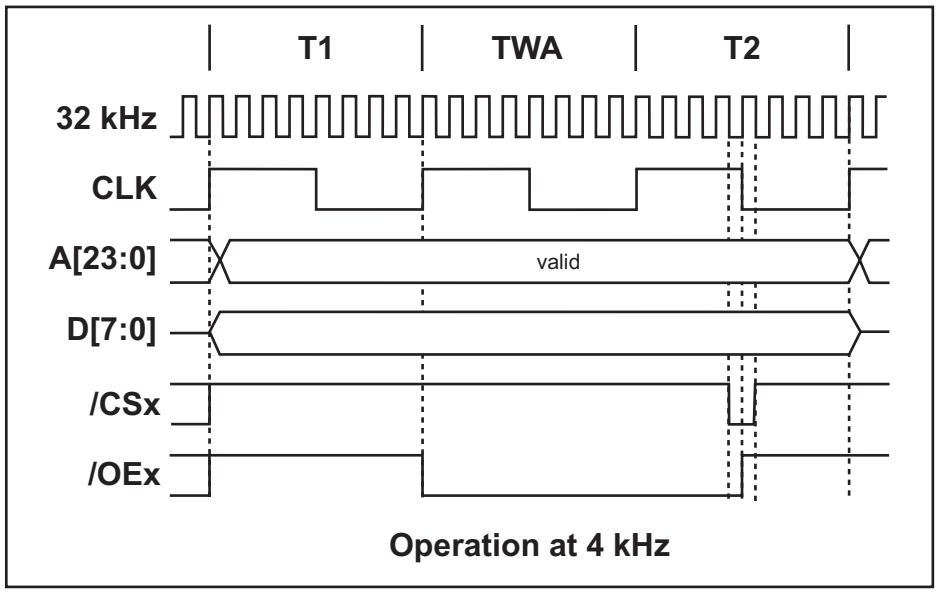


Figure B-17. Short Chip Select Timing: 4 kHz, Write Operation

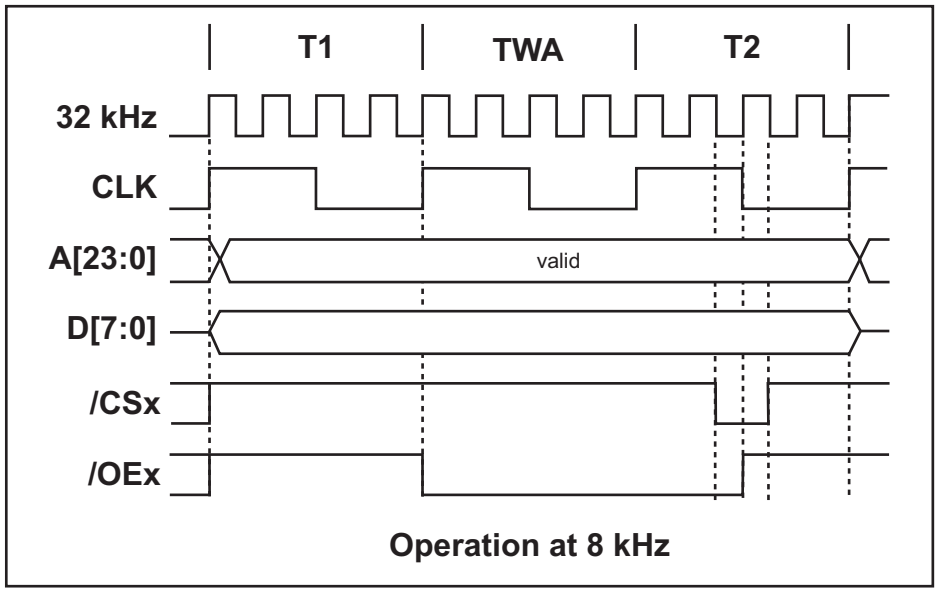


Figure B-18. Short Chip Select Timing: 8 kHz, Write Operation

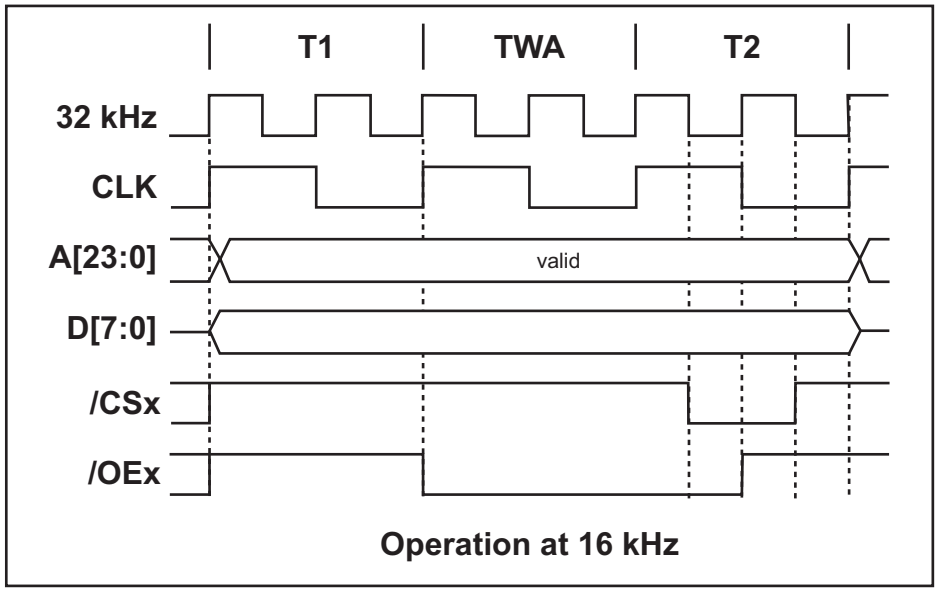


Figure B-19. Short Chip Select Timing: 16 kHz, Write Operation

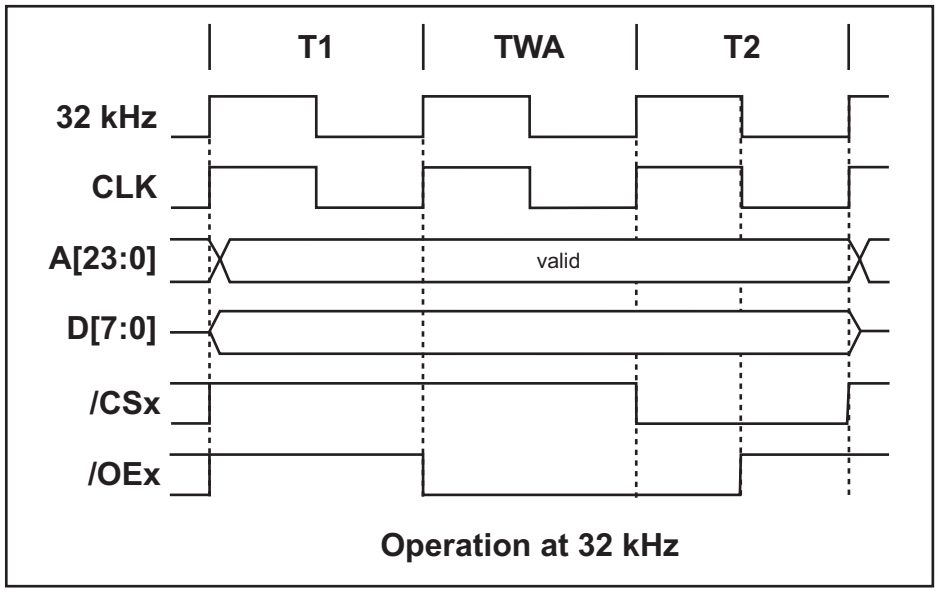


Figure B-20. Short Chip Select Timing: 32 kHz, Write Operation

B.1.13 Pulse Width Modulator Improvements

Several new features have been added to the pulse width modulator. First, a new PWM interrupt can be set up to be requested on every PWM cycle, every other cycle, every fourth cycle, or every eighth cycle. The setup for this interrupt is done in the PWL0R and PWL1R registers.

Options are available to suppress the PWM output for seven-of-eight, three-of-four and one-of-two iterations of the PWM counter The one-of-eight option works nicely with R/C servos, which require a 1 ms to 2 ms pulse width and a 20 ms period. This option gives the full resolution for the pulse width while still meeting the period requirements. The one-of-four and one-of-two options can be used to create more virtual PWM channels using software to multiplex the PWM outputs. There is a separate option to only generate an interrupt during the active iteration of the PWM count. The timing is shown below.

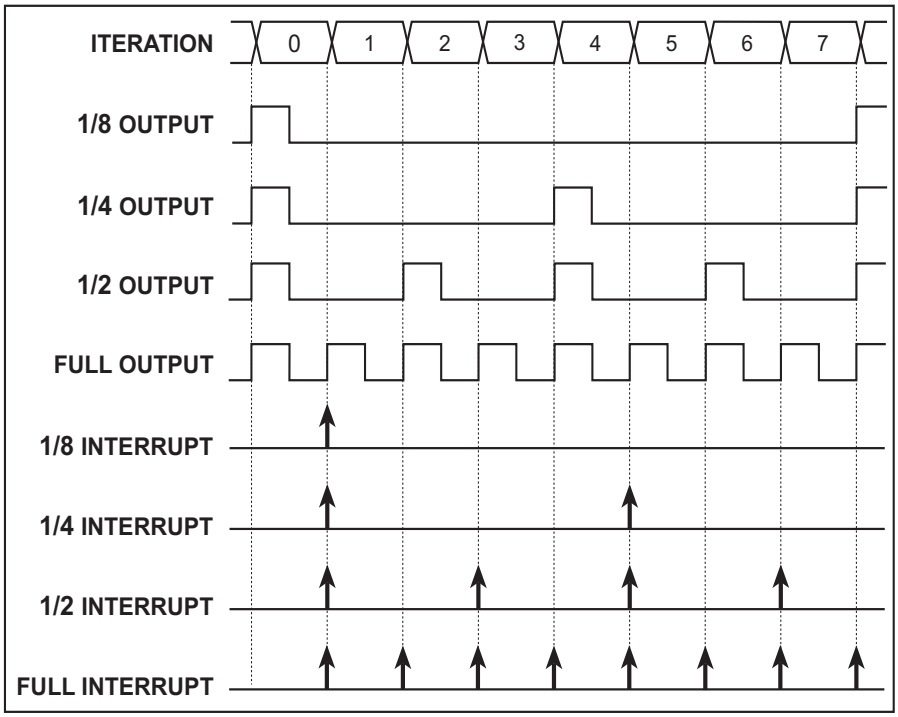


Figure B-21. PWM Interrupt and Output Timing

B.1.14 Quadrature Decoder Improvements

The Quadrature Decoder counters can now be expanded to 10 bits instead of 8 bits. This is controlled by bit 5 in QDCR. The additional two bits can be read in the QDCxHR registers. Reading the lower 8 bits of the Quadrature Decoder latches the upper 2 bits, which can then be read at any time afterwards. The latch is cleared when the upper 2 bits are read, and subsequent reads of these upper two bits will return the current counter value until they are latched again by another read of the lower 8 bits.

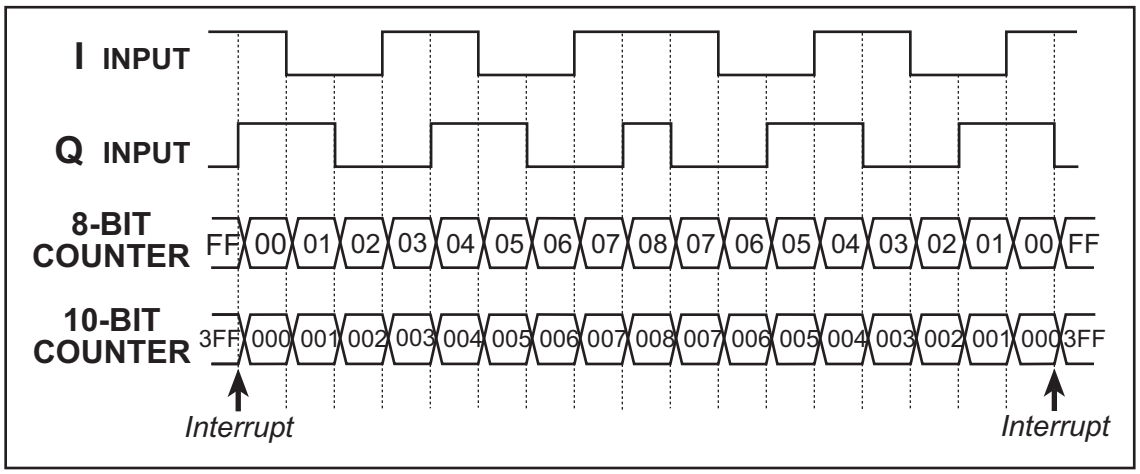


Figure B-22. Quadrature Decode, 8-bit and 10-bit Counter Timing

B.2 Pins with Alternate Functions

The Rabbit 3000A provides greater flexibility for multiplexing I/O functions to other pins. The following alternate connections were introduced in the Rabbit 3000A for these peripherals, and are indicated by an asterisk in Appendix A.

- Slave port CS

  /ASCS: Alternate slave port chip select input

- Serial Ports E/F

  ARXE: Alternate Serial Port E receive

  ARCLKE: Alternate Serial Port E receive clock (HDLC)

  ARXF: Alternate Serial Port F receive

  ARCLKF: Alternate Serial Port F receive clock (HDLC)

- PWM outputs

  APWM3: Alternate PWM output, bit 3

  APWM2: Alternate PWM output, bit 2

  APWM1: Alternate PWM output, bit 1

  APWM0: Alternate PWM output, bit 0

INDEX

I

L

M

O

P

Q

R

S

T

W