IP3023™ Wireless Network Processor

Eight-Way Multithreaded Processor Optimized for Network Connectivity

1.0 Product Highlights

The IP3023™ wireless network processor is a revolutionary new platform from Ubicom designed to provide highly integrated solutions for applications at the "edge" of Internet connectivity, including 802.11a/b/g access points, routers, hot spots, bridges, gateways, and a wide variety of embedded networked client solutions. The IP3023 is optimized for efficient network processing in embedded solutions. Its development has led to the definition of a new microprocessor architecture: Multithreaded Architecture for Software I/O (MASI). Many MASI concepts were pioneered in the Ubicom IP2000™ family of processors, but the IP3023 dramatically extends those techniques by introducing hardware support for multiple threads operating with no context switching overhead, as well as three-operand and memory-to-memory operations.

The IP3023 is a 250- or 325-MIPS 32-bit CPU supporting eight-way multithreaded operation. It provides for up to eight real-time tasks to execute in a completely deterministic fashion. In essence, the IP3023 supports running a different thread on every clock, but without the overhead for context switching typical with traditional microprocessor architectures. To the system designer, the IP3023 appears as if there were eight processors on the chip.

Key Features:

- 32-bit Mutithreaded CPU, in 250 MIPS and 325 MIPS versions
- IP3023 is optimized for wireless networking
  - Eight-way simultaneous multithreading
  - Deterministic execution on all threads
  - Zero overhead full context switching
  - Programmable MIPS per thread
  - Optimized ISA for packet processing
    - Memory-to-memory architecture, powerful addressing modes
    - Small, fast instruction set, strong bit manipulation
    - Reduced code size vs. RISC CPUs
- On-chip program and data memory
  - Eliminates cache miss penalties
  - 256 KB (64K x 32) of program SRAM
  - 64 KB (16K x 32) of data SRAM
- Highly configurable I/O support
  - Many combinations of software I/O:
    - Utopia, PCMCIA, IDE/ATAPI
    - PCM Highway, UART, SPI, I^{2}C
    - 32-bit 802.11a/g radios interface
  - Up to 4 MII ports of 10/100 PHY
  - Two SerDes for fast serial I/O:
    - 10Base-T (MAC/PHY), USB, GPSI
    - SPI, UART, two-wire serial, BlueRF
  - High-speed GPSI port
- Additional key hardware
  - True random number generator for software-implemented encryption/security (32-bit seed)
  - Fixed-point MAC (16x16+48-bit, 250/325 MMACs) for voice/audio codecs, other signal processing tasks
- Independent I/O and core CPU clocking
  - Separate phase-locked loops (PLLs)
  - Programmable multipliers & dividers
  - Single low-cost crystal (10–20 MHz)

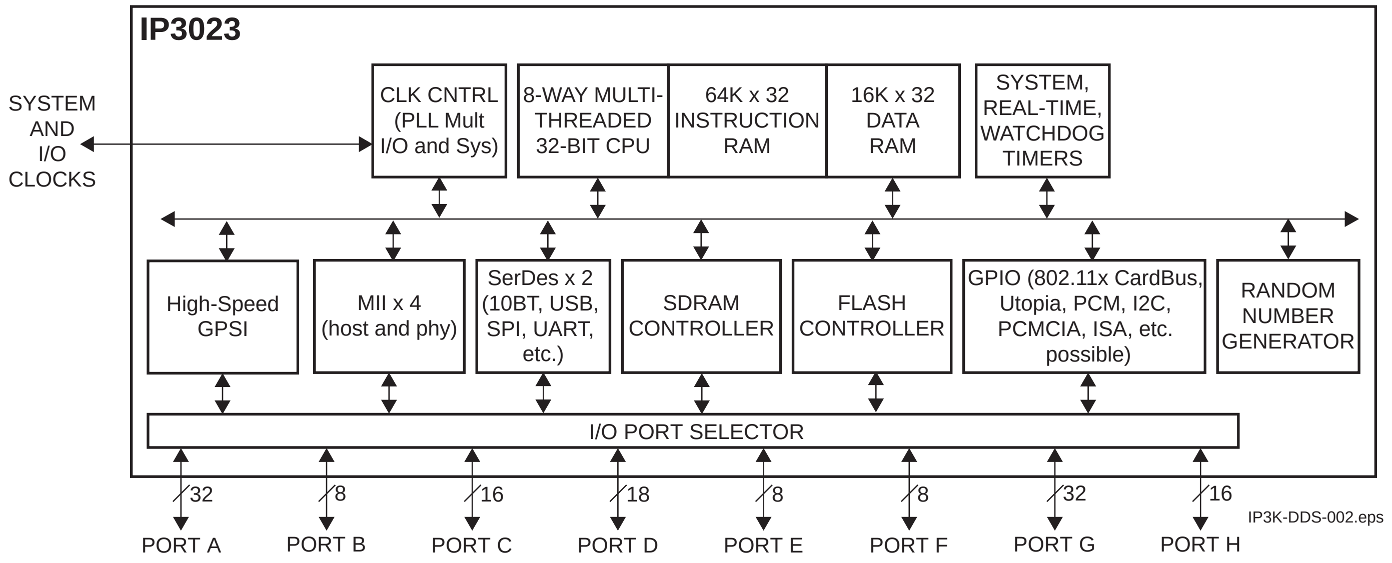


Figure 1-1 IP3023 Block Diagram

The multithreaded and deterministic nature of the IP3023 processor provides for integration of numerous functions on chip – some with on-chip hardware assist and some entirely in software – as threads, including the ability to support interfaces such as 10/100 MII and 10Base-T Ethernet MAC/PHY, USB, GPSI, Utopia, PCMCIA, IDE, PCM Highway, and CardBus/Mini PCI interface specific for 802.11a/g wireless radios. This yields both a high degree of flexibility and reduces die size, as it eliminates the need for many on-chip dedicated hardware blocks for specific functions.

The IP3023 employs a three-operand and memory-to-memory architecture, utilizing on-chip program and data memory support. This scheme enables highly efficient data movement and processing on data. The result is that the IP3023 is designed to support packet processing and transfers at wire speeds, eliminating the need for caches and large data buffers typically found in use with traditional RISC-based microprocessors.

To further optimize the IP3023 for networking infrastructure and embedded client solutions, the processor includes several key hardware support blocks, including true random number generator and fixed-point multiply/accumulate (MAC) units. The random number generator facilitates robust software implementation of common encryption/security protocols critical to the continued growth of wireless networking. The MAC unit supports implementation of voice/audio codecs and other signal processing tasks.

1.1 Additional Features

IP3023 Wireless Network Processor Capabilities

Foundation for Highly Flexible Connectivity Solution

- Performance: 250 MIPS @ 250MHz,
  325MIPS @ 325MHz
- 250/325 MMACs performance from MAC unit with fixed point 16x16 multiply and 48-bit accumulator.
- On-chip dual-ported SRAM data memory.
- On-chip SRAM program memory.
- On-chip hardware for zero overhead instruction level context switch for multithreading.
- In-system programming of external flash.

Multiple Networking Protocols and Physical Layer Support Hardware

- Two full-duplex serializer/deserializer (SerDes) channels.
  - Flexible to support 10Base-T, GPSI, SPI, UART, USB protocols.
  - One on-chip PHY function for 10Base-T Ethernet.
- Four MII Ports, each can operate in host or phy mode.

Memory

- 256 KB (64K x 33) on-chip program SRAM with 1-bit parity.
- 64 KB (16K x 32) on-chip dual port data SRAM.
- Up to 4 MB (4M x 8) off-chip flash support.
- Up to 64 MB (32M x 16) off-chip PC 66/100/133 SDRAM support.

IP3023 Wireless Network Processor Features

- 32-bit data and instruction paths (fixed instruction width).
- Instructions execute at the rate of one per clock cycle.
- Eight-way instruction level multithreading with support for both hard real time and non-real-time thread priority scheduling.
- Multiply and multiply/accumulate (MAC) instructions, where MAC uses a 48-bit accumulator.
- Special purpose CRC instruction for CRC generation/checking and encryption.
- Sixteen general-purpose 32-bit registers per thread (128 total).
- Eight 32-bit address registers per thread (64 total).
- High instruction code density.

General-Purpose Hardware Peripherals

- True random number generator (32-bit seed number).
- One 32-bit system timer synchronous with system clock with eight compare registers.
- One 32-bit real-time timer (one mode of the Multipurpose Timer) with constant clock frequency.
- Watchdog 32-bit timer with constant clock frequency.
- Power-on reset circuit.
- Eight external interrupt inputs mapped to I/O ports.
- Two programmable output clocks.

Sophisticated Power and Frequency/Clock Management Support

- Operating voltage from 1.14V to 1.26V.
- Single clock input with 10–20 MHz crystal clock input support.
- Two on-chip PLLs: one for processor clock, one for serial I/O.
- Core clock using a selectable on-chip divider.
- Software CPU speed control for power saving.
- Power-on reset (POR) logic.
- Auxiliary I/O clock input for serial I/O PLL.

Flexible I/O

- 138 I/O pins in 228-pin 17x17mm BGA option.
- Eight configurable multifunction I/O ports.
- 3.3V symmetric CMOS output drive.
- 5V-tolerant I/O.

Support for In-System Debug and Configuration

- Customer application program updatable.
  - Run-time self programming.
- On-chip in-system debugging support interface.
- Debugging at full IP3023 operating speed.

Complete Software Development Environment

The IP3023 wireless network processor is capable of supporting the following functions in software. See a Ubicom sales representative for availability and schedule.

- ipOS™ operating system
- ipModule™ software – pre-built connectivity modules – complete software solutions and platform for wireless routers, access points, bridges, print servers, network cameras, and many other embedded network client applications. Includes core networking stack, numerous wired and wireless PHY layers and interfaces, plus application layer ipModules for WebServer, DNS, SMTP email, SNMP remote management, DHCP client/server, and NAT routing, to name a few.
- Red Hat® GNUPro® tools.
  - Includes GCC ANSI C compiler and assembler, linker, utilities, and GNU debugger.
- Configuration tool.
  - Integrated tool to support rapid development efforts.
- Ubicom's Unity™ integrated development environment (IDE).
  - Includes editor, project manager, graphical user interface to GNU debugger, device programmer, ipModule configuration tool, and profiler.
- Profiler for performance tuning.

1.2 I/O Port Mapping

The IP3023 has highly configurable port mapping. Port A, for example, supports the external flash, but can also be shared with the SDRAM controller and a PCMCIA interface when combined with Port B. Some of the ports share a dedicated hardware function. Port E, for example, shares a part of an MII port, one of the on-board SerDes units, or part of a Utopia bus implementation in software. All ports (except Port A) can be used as GPIO ports, giving up to 106 GPIO pins. GPIO ports are used to create virtual I/O ports to control UTOPIA, 802.11a/b/g, PCM Highway, and other popular interfaces. Table 1-1 shows how the IP3023 I/O ports are shared and shows possible I/O port mappings for three different applications.

1.3 Architecture

1.3.1 CPU

The CPU is a general-purpose 32-bit pipelined processor. The CPU implements multithreading in hardware and supports the execution of deterministic hard real-time (HRT) threads. Up to eight simultaneous threads are supported in hardware. Code for the processing core is written in C as well as in assembly language.

1.3.2 CPU Instruction Memory

The instruction memory for the main processor is implemented as a single-port (256 KB with parity, 64K x 32) SRAM. This RAM is able to supply the main processor with one instruction access per clock.

1.3.3 CPU Data Memory

The data memory for the main processor is 64 KB (16K x 32) SRAM. It is able to perform one read and one write per clock, in support of the three-operand and memory-to-memory instruction set architecture.

1.3.4 Clocks, Frequency, and Timers

A single clock input (crystal, 10–20 MHz) is used to source multiple subsystems and peripherals in the IP3023. This clock source is fed into independent PLLs for generating a system clock and a serial I/O clock. Alternatively, the PLLs can be bypassed, and the 10–20 MHz clock input can be used directly. The PLLs are capable of generating up to a 250 MHz core clock from the 10–20 MHz input signal, or up to 325MHz on the 325MHz rated version.

This 10–20 MHz input is also fed into a real-time clock (RTC) timer portion of the multipurpose timer, which can be used to maintain an accurate time base in a system.

Table 1-1 I/O Ports and Example Configurations

I/O Port	Port Width (bits)	Actual Hardware I/O Support	Dual / Multimode Access Point	Print Server Bridge	Networked Embedded Devices
Port A	32	Flash	Flash	Flash	Flash
Port B	8	SDRAM	GPIO / SDRAM (opt.)	GPIO / SDRAM (opt.)	GPIO / SDRAM (opt.)
Port C	16	MII	GPIO	GPIO	GPIO
Port D	18	MII	MII	MII	MII
Port E	8	SerDes or MII (1/2) or Hi-Speed GPSI.	GPOI	GPIO	GPIO
Port F	8	SerDes or MII (1/2)	GPIO	USB	GPIO
Port G	32	GPIO	802.11 (a and g)	802.11 (a or g)	802.11 (a or g)
Port H	16	MII			

1.3.5 Reduced Power Operation

The IP3023 can be configured for lower power operation by varying the frequency of operation and clock source. These lower power modes include:

- *Reducing the clock frequency from the system clock PLL.* The clock circuit of the IP3023 includes a run-time controllable CPU clock, which allows the developer to reduce operating frequency (see Figure 3-3).
- *Turning off system clock PLL.* System clocking runs directly from the 10–20 MHz clock input.

1.3.6 Interrupts

The IP3023 provides a flexible interrupt structure. Real-time interrupts are individually assigned to independent threads. An interrupt awakens the corresponding thread, if it was waiting, and the thread handles the interrupt with the priority and processor cycles assigned to the thread. If needed, the structure of a traditional interrupt service routine (ISR) can be emulated.

1.3.7 Reset

The following sources are capable of causing a chip reset:

- Power-on
- Debug port
- Watchdog timer (one mode of the multipurpose timer)
- Parity error in on-chip instruction memory
- External reset ($\overline{\text{RST}}$ pin)

1.3.8 Programming and Debugging

The IP3023 device has advanced in-system programming and debug support on-chip. This unobtrusive capability is provided through a Debug Interface. There is no need for a bond-out chip for software development. This eliminates concerns about differences in electrical characteristics between a bond-out chip and the actual chip used in the target application. Designers can test and revise code on the same part used in the actual application.

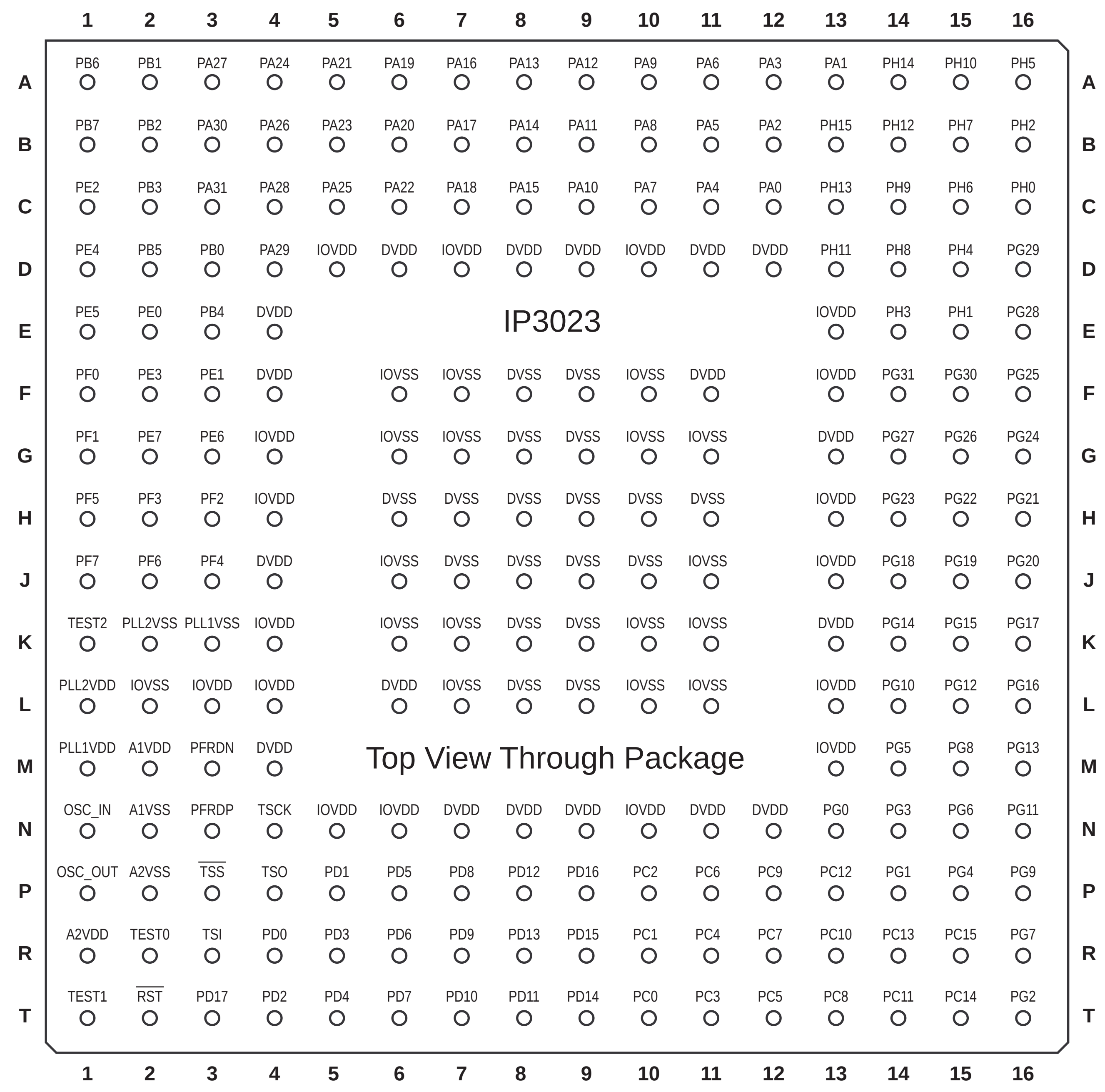
Ubicom provides the complete Red Hat GNUPro tools, including C compiler, assembler, linker, utilities, and GNU debugger. In addition, Ubicom offers an integrated graphical development environment which includes an editor, project manager, graphical user interface for the GNU debugger, device programmer, ipModule configuration tool, and profiler.

1.3.9 Other Supported Functions

- *Random-number generator.* The IP3023 includes an on-chip hardware true random number generator. On-chip random noise generates random bits which are accumulated in a hardware 32-bit linear feedback shift register (LFSR). This function can be used to seed a software random number generator or to generate per-session cryptography keys.
- *Boot from external flash.* Unlike the IP2000 family processors, there is no on-chip flash. At start-up, the IP3023 is configured to execute instructions from its external flash port.

2.0 Pin Definitions

2.1 228-Pin BGA



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	PB6	PB1	PA27	PA24	PA21	PA19	PA16	PA13	PA12	PA9	PA6	PA3	PA1	PH14	PH10	PH5
B	PB7	PB2	PA30	PA26	PA23	PA20	PA17	PA14	PA11	PA8	PA5	PA2	PH15	PH12	PH7	PH2
C	PE2	PB3	PA31	PA28	PA25	PA22	PA18	PA15	PA10	PA7	PA4	PA0	PH13	PH9	PH6	PH0
D	PE4	PB5	PB0	PA29	IOVDD	DVDD	IOVDD	DVDD	DVDD	IOVDD	DVDD	DVDD	PH11	PH8	PH4	PG29
E	PE5	PE0	PB4	DVDD									IOVDD	PH3	PH1	PG28
F	PF0	PE3	PE1	DVDD		IOVSS	IOVSS	DVSS	DVSS	IOVSS	DVDD		IOVDD	PG31	PG30	PG25
G	PF1	PE7	PE6	IOVDD		IOVSS	IOVSS	DVSS	DVSS	IOVSS	IOVSS		DVDD	PG27	PG26	PG24
H	PF5	PF3	PF2	IOVDD		DVSS	DVSS	DVSS	DVSS	DVSS	DVSS		IOVDD	PG23	PG22	PG21
J	PF7	PF6	PF4	DVDD		IOVSS	DVSS	DVSS	DVSS	DVSS	IOVSS		IOVDD	PG18	PG19	PG20
K	TEST2	PLL2VSS	PLL1VSS	IOVDD		IOVSS	IOVSS	DVSS	DVSS	IOVSS	IOVSS		DVDD	PG14	PG15	PG17
L	PLL2VDD	IOVSS	IOVDD	IOVDD		DVDD	IOVSS	DVSS	DVSS	IOVSS	IOVSS		IOVDD	PG10	PG12	PG16
M	PLL1VDD	A1VDD	PFRDN	DVDD									IOVDD	PG5	PG8	PG13
N	OSC_IN	A1VSS	PFRDP	TSCK	IOVDD	IOVDD	DVDD	DVDD	DVDD	IOVDD	DVDD	DVDD	PG0	PG3	PG6	PG11
P	OSC_OUT	A2VSS	$\overline{\text{TSS}}$	TSO	PD1	PD5	PD8	PD12	PD16	PC2	PC6	PC9	PC12	PG1	PG4	PG9
R	A2VDD	TEST0	TSI	PD0	PD3	PD6	PD9	PD13	PD15	PC1	PC4	PC7	PC10	PC13	PC15	PG7
T	TEST1	$\overline{\text{RST}}$	PD17	PD2	PD4	PD7	PD10	PD11	PD14	PC0	PC3	PC5	PC8	PC11	PC14	PG2

Figure 2-1 IP3023 BGA Pin Definition (Top View)

Table 2-1 Pin Assignments (sorted by pin number)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
A1	PB6	D10	IOVDD	J1	PF7	N8	DVDD
A2	PB1	D11	DVDD	J2	PF6	N9	DVDD
A3	PA27	D12	DVDD	J3	PF4	N10	IOVDD
A4	PA24	D13	PH11	J4	DVDD	N11	DVDD
A5	PA21	D14	PH8	J6	IOVSS	N12	DVDD
A6	PA19	D15	PH4	J7	DVSS	N13	PG0
A7	PA16	D16	PG29	J8	DVSS	N14	PG3
A8	PA13	E1	PE5	J9	DVSS	N15	PG6
A9	PA12	E2	PE0	J10	DVSS	N16	PG11
A10	PA9	E3	PB4	J11	IOVSS	P1	OSC_OUT
A11	PA6	E4	DVDD	J13	IOVDD	P2	A2VSS
A12	PA3	E13	IOVDD	J14	PG18	P3	$\overline{\text{TSS}}$
A13	PA1	E14	PH3	J15	PG19	P4	TSO
A14	PH14	E15	PH1	J16	PG20	P5	PD1
A15	PH10	E16	PG28	K1	TEST2	P6	PD5
A16	PH5	F1	PF0	K2	PLL2VSS	P7	PD8
B1	PB7	F2	PE3	K3	PLL1VSS	P8	PD12
B2	PB2	F3	PE1	K4	IOVDD	P9	PD16
B3	PA30	F4	DVDD	K6	IOVSS	P10	PC2
B4	PA26	F6	IOVSS	K7	IOVSS	P11	PC6
B5	PA23	F7	IOVSS	K8	DVSS	P12	PC9
B6	PA20	F8	DVSS	K9	DVSS	P13	PC12
B7	PA17	F9	DVSS	K10	IOVSS	P14	PG1
B8	PA14	F10	IOVSS	K11	IOVSS	P15	PG4
B9	PA11	F11	DVDD	K13	DVDD	P16	PG9
B10	PA8	F13	IOVDD	K14	PG14	R1	A2VDD
B11	PA5	F14	PG31	K15	PG15	R2	TEST0
B12	PA2	F15	PG30	K16	PG17	R3	TSI
B13	PH15	F16	PG25	L1	PLL2VDD	R4	PD0
B14	PH12	G1	PF1	L2	IOVSS	R5	PD3
B15	PH7	G2	PE7	L3	IOVDD	R6	PD6
B16	PH2	G3	PE6	L4	IOVDD	R7	PD9
C1	PE2	G4	IOVDD	L6	DVDD	R8	PD13
C2	PB3	G6	IOVSS	L7	IOVSS	R9	PD15
C3	PA31	G7	IOVSS	L8	DVSS	R10	PC1
C4	PA28	G8	DVSS	L9	DVSS	R11	PC4
C5	PA25	G9	DVSS	L10	IOVSS	R12	PC7
C6	PA22	G10	IOVSS	L11	IOVSS	R13	PC10
C7	PA18	G11	IOVSS	L13	IOVDD	R14	PC13
C8	PA15	G13	DVDD	L14	PG10	R15	PC15
C9	PA10	G14	PG27	L15	PG12	R16	PG7
C10	PA7	G15	PG26	L16	PG16	T1	TEST1
C11	PA4	G16	PG24	M1	PLL1VDD	T2	$\overline{\text{RST}}$
C12	PA0	H1	PF5	M2	A1VDD	T3	PD17
C13	PH13	H2	PF3	M3	PFRDN	T4	PD2
C14	PH9	H3	PF2	M4	DVDD	T5	PD4
C15	PH6	H4	IOVDD	M13	IOVDD	T6	PD7
C16	PH0	H6	DVSS	M14	PG5	T7	PD10
D1	PE4	H7	DVSS	M15	PG8	T8	PD11
D2	PB5	H8	DVSS	M16	PG13	T9	PD14
D3	PB0	H9	DVSS	N1	OSC_IN	T10	PC0
D4	PA29	H10	DVSS	N2	A1VSS	T11	PC3
D5	IOVDD	H11	DVSS	N3	PFRDP	T12	PC5
D6	DVDD	H13	IOVDD	N4	TSCK	T13	PC8
D7	IOVDD	H14	PG23	N5	IOVDD	T14	PC11
D8	DVDD	H15	PG22	N6	IOVDD	T15	PC14
D9	DVDD	H16	PG21	N7	DVDD	T16	PG2

Table 2-2 Pin Assignments (sorted by signal name)

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
A1VDD	M2	IOVSS	F6	PB6	A1	PG3	N14
A1VSS	N2	IOVSS	F7	PB7	B1	PG4	P15
A2VDD	R1	IOVSS	F10	PC0	T10	PG5	M14
A2VSS	P2	IOVSS	G6	PC1	R10	PG6	N15
DVDD	D6	IOVSS	G7	PC2	P10	PG7	R16
DVDD	D8	IOVSS	G10	PC3	T11	PG8	M15
DVDD	D9	IOVSS	G11	PC4	R11	PG9	P16
DVDD	D11	IOVSS	J6	PC5	T12	PG10	L14
DVDD	D12	IOVSS	J11	PC6	P11	PG11	N16
DVDD	E4	IOVSS	K6	PC7	R12	PG12	L15
DVDD	F4	IOVSS	K7	PC8	T13	PG13	M16
DVDD	F11	IOVSS	K10	PC9	P12	PG14	K14
DVDD	G13	IOVSS	K11	PC10	R13	PG15	K15
DVDD	J4	IOVSS	L2	PC11	T14	PG16	L16
DVDD	K13	IOVSS	L7	PC12	P13	PG17	K16
DVDD	L6	IOVSS	L10	PC13	R14	PG18	J14
DVDD	M4	IOVSS	L11	PC14	T15	PG19	J15
DVDD	N7	OSC_IN	N1	PC15	R15	PG20	J16
DVDD	N8	OSC_OUT	P1	PD0	R4	PG21	H16
DVDD	N9	PA0	C12	PD1	P5	PG22	H15
DVDD	N11	PA1	A13	PD2	T4	PG23	H14
DVDD	N12	PA2	B12	PD3	R5	PG24	G16
DVSS	F8	PA3	A12	PD4	T5	PG25	F16
DVSS	F9	PA4	C11	PD5	P6	PG26	G15
DVSS	G8	PA5	B11	PD6	R6	PG27	G14
DVSS	G9	PA6	A11	PD7	T6	PG28	E16
DVSS	H6	PA7	C10	PD8	P7	PG29	D16
DVSS	H7	PA8	B10	PD9	R7	PG30	F15
DVSS	H8	PA9	A10	PD10	T7	PG31	F14
DVSS	H9	PA10	C9	PD11	T8	PH0	C16
DVSS	H10	PA11	B9	PD12	P8	PH1	E15
DVSS	H11	PA12	A9	PD13	R8	PH2	B16
DVSS	J7	PA13	A8	PD14	T9	PH3	E14
DVSS	J8	PA14	B8	PD15	R9	PH4	D15
DVSS	J9	PA15	C8	PD16	P9	PH5	A16
DVSS	J10	PA16	A7	PD17	T3	PH6	C15
DVSS	K8	PA17	B7	PE0	E2	PH7	B15
DVSS	K9	PA18	C7	PE1	F3	PH8	D14
DVSS	L8	PA19	A6	PE2	C1	PH9	C14
DVSS	L9	PA20	B6	PE3	F2	PH10	A15
IOVDD	D5	PA21	A5	PE4	D1	PH11	D13
IOVDD	D7	PA22	C6	PE5	E1	PH12	B14
IOVDD	D10	PA23	B5	PE6	G3	PH13	C13
IOVDD	E13	PA24	A4	PE7	G2	PH14	A14
IOVDD	F13	PA25	C5	PF0	F1	PH15	B13
IOVDD	G4	PA26	B4	PF1	G1	PLL1VDD	M1
IOVDD	H4	PA27	A3	PF2	H3	PLL1VSS	K3
IOVDD	H13	PA28	C4	PF3	H2	PLL2VDD	L1
IOVDD	J13	PA29	D4	PF4	J3	PLL2VSS	K2
IOVDD	K4	PA30	B3	PF5	H1	$\overline{\text{RST}}$	T2
IOVDD	L3	PA31	C3	PF6	J2	TEST0	R2
IOVDD	L4	PB0	D3	PF7	J1	TEST1	T1
IOVDD	L13	PB1	A2	PFRDN	M3	TEST2	K1
IOVDD	M13	PB2	B2	PFRDP	N3	TSCK	N4
IOVDD	N5	PB3	C2	PG0	N13	TSI	R3
IOVDD	N6	PB4	E3	PG1	P14	TSO	P4
IOVDD	N10	PB5	D2	PG2	T16	$\overline{\text{TSS}}$	P3

2.2 Pin Descriptions

Type Codes: I = Digital Input, AI = Analog Input, O/DO = Digital Output, HiZ = High Impedance, P = Power,

PU = On-Chip Pullup, PD = On-Chip Pulldown, ST = Schmitt Trigger, NS = Non-Slew Rate Limited

Table 2-3 Pin Descriptions

Name	Type	Sink @ 3.3V IOVDD	Source @ 3.3V IOVDD	Description
A1VDD	P 1.2V			VDD for analog blocks: power-on circuit, crystal oscillator and rc-oscillator
A1VSS	P 0V			VSS for analog blocks: power-on circuit, crystal oscillator and rc-oscillator
A2VDD	P 1.2V			VDD for squelch block only
A2VSS	P 0V			VSS for squelch block only
DVDD	P 1.2V			VDD for digital core
DVSS	P 0V			VSS for digital core
IOVDD	P 3.3V			VDD for I/Os
IOVSS	P 0V			VSS for I/Os
OSC_IN	AI			Crystal clock input
OSC_OUT	O/HiZ			Crystal clock output
PFRDN	AI			Port F Ethernet RXN
PFRDP	AI			Port F Ethernet RXP
PLL1VDD	P 1.2V			VDD for core clock PLL
PLL1VSS	P 1.2V			VSS for core clock PLL
PLL2VDD	P 1.2V			VDD for serial I/O clock PLL
PLL2VSS	P 1.2V			VSS for serial I/O clock PLL
PA[31:0]	I/O	6 mA	8 mA	Port A. Refer to Table 2-4.
PB[7,5,3:0]	I/O	6 mA	8 mA	Port B. Refer to Table 2-5.
PB6	I/O, NS	16 mA	16 mA	Port B. Refer to Table 2-5. See Note 2 below.
PB4	I/O	6 mA	8 mA	SDRAM enable – A pullup must be connected to this pin if an external SRAM is used; if not, a pulldown should be used. See Note 2.
PC[15:0]	I/O	6 mA	8 mA	Port C. Refer to Table 2-6.
PD17	I/O, ST	6 mA	8 mA	Port D. Auxiliary I/O clock input, refer to Table 2-7.
PD[16:0]	I/O	6 mA	8 mA	Port D. Refer to Table 2-7.
PE[7:0]	I/O	6 mA	8 mA	Port E. Refer to Table 2-8.
PF7	I/O	6 mA	8 mA	Port F. Refer to Table 2-9.
PF[6:5]	I/O	16 mA	24 mA	Port F. Refer to Table 2-9.
PF[4:0]	I/O	6 mA	8 mA	Port F. Refer to Table 2-9.
PG[31:0]	I/O	6 mA	8 mA	Port G. Refer to Table 2-10.
PH[15:0]	I/O	6 mA	8 mA	Port H. Refer to Table 2-11.

Table 2-3 Pin Descriptions

Name	Type	Sink @ 3.3V IOVDD	Source @ 3.3V IOVDD	Description
$\overline{RST}$	I/ST/PU			Assert to 0 for chip reset. See Note 1.
TEST0, TEST1, TEST2	I, PD			Test mode pins. Connect to Vss. See Note 1.
TSCK	I/ST/PD			Debug Interface Clock (used only for in-system programming and debug).
$\overline{TSS}$	I/ST/PU			Debug Interface Slave Select (used only for in-system programming and debug). See Note 1.
TSI	I/ST/PU			Debug Interface Serial Data Input (used only for in-system programming and debug).
TSO	O/HiZ	6 mA	8 mA	Debug Interface Serial Data output (used only for in-system programming and debug; high Z unless TSS low)
Note 1:	Ubicom recommends not relying on internal pullup or pulldown.			
Note 2:	If the PB4 pin is pulled high at power up, the PB6 pin will output at least 2 clocks (for external SDRAM) at the frequency of the OSC_IN pin until the reset holdoff time expires, and then PB6 will behave normally (tri-stated until programmed by software). If the PB4 pin is pulled low during power up, the PB6 pin will remain tri-stated until programmed by software.			

2.3 I/O Ports Signal Maps

The eight I/O ports are designated Port A, Port B, ... , Port H. Every port is capable of multiple functions. Programs select the function of a port by programming the port's function select register. Behavior of each I/O port's signals depends on the function selected for that port. Table 2-4 through Table 2-11 show the signal assignments for each function of each port. Refer also to Section 5.0 on page 43 for more detail and for explanations of terms.

Table 2-4 Port A Signal Map

Port Bit # PA[*n*]	Function 0 (Flash)	Function 1 (SDRAM)	Function 3 (GPIO)
12:0	ADDR [12:0]	ADDR [12:0]	GPIO
13	ADDR [13]	BA [0]	GPIO
14	ADDR [14]	BA [1]	GPIO
15	ADDR [15]	DQM	GPIO
16	ADDR [16]	DATA [0]	GPIO
17	ADDR [17]	DATA [1]	GPIO
18	ADDR [18]	DATA [2]	GPIO
19	ADDR [19]	DATA [3]	GPIO
20	ADDR [20]	DATA [4]	GPIO
21	ADDR [21]	DATA [5]	GPIO
22	$\overline{OE}$	DATA [6]	GPIO
23	$\overline{WE}$	DATA [7]	GPIO
24	DATA [0]	DATA [8]	GPIO
25	DATA [1]	DATA [9]	GPIO
26	DATA [2]	DATA [10]	GPIO
27	DATA [3]	DATA [11]	GPIO
28	DATA [4]	DATA [12]	GPIO
29	DATA [5]	DATA [13]	GPIO
30	DATA [6]	DATA [14]	GPIO
31	DATA [7]	DATA [15]	GPIO

Table 2-5 Port B Signal Map

Port Bit # PB[*n*]	Function 0 (Flash)	Function 1 (SDRAM + Flash + Clock)	Function 2 (GPIO)
0	GPIO	$\overline{\text{WE}}$	GPIO
1	GPIO	$\overline{\text{RAS}}$	GPIO
2	GPIO	$\overline{\text{CAS}}$	GPIO
3	GPIO	$\overline{\text{CS}}$	GPIO
4	GPIO	CKE	GPIO
5	GPIO	SD_CLK_IN	GPIO
6	GPIO	CLK_OUT	GPIO
7	$\overline{\text{FCE}}$*	$\overline{\text{FCE}}$*	GPIO

* A pullup should be placed on the $\overline{\text{FCE}}$ (Flash Chip Enable) pin so that the Flash isn't enabled while the IP3023 is in reset (while the IP3023 is floating the $\overline{\text{FCE}}$ pin).

Table 2-6 Port C Signal Map

Port Bit # PC[*n*]	Func 0 (GPIO)	Func 1 (MII)
0	GPIO	CRS
1	GPIO	COL
2	GPIO	TXD [3]
3	GPIO	TXD [2]
4	GPIO	TXD [1]
5	GPIO	TXD [0]
6	GPIO	TX_EN
7	GPIO	TX_CLK
8	GPIO	TX_ER
9	GPIO	RX_ER
10	GPIO	RX_CLK
11	GPIO	RX_DV
12	GPIO	RXD [0]
13	GPIO	RXD [1]
14	GPIO	RXD [2]
15	GPIO	RXD [3]

Table 2-7 Port D Signal Map

Port Bit # PD[*n*]	Function 0 (GPIO)	Function 1 (MII)
0	GPIO	CRS
1	GPIO	COL
2	GPIO	TXD [3]
3	GPIO	TXD [2]
4	GPIO	TXD [1]
5	GPIO	TXD [0]
6	GPIO	TX_EN
7	GPIO	TX_CLK
8	GPIO	TX_ER
9	GPIO	RX_ER
10	GPIO	RX_CLK
11	GPIO	RX_DV
12	GPIO	RXD [0]
13	GPIO	RXD [1]
14	GPIO	RXD [2]
15	GPIO	RXD [3]
16	GPIO	GPIO
17*	GPIO	GPIO

* Auxiliary serial I/O clock input

Table 2-8 Port E Signal Map

Port Bit # PE[*n*]	Func 0 (GPIO)	Func 1 (SerDes)	Func 2 (1/2 MII)	Func 3 GPSI
0	GPIO	RXD	CRS	RxD
1	GPIO	RXM	COL	COL
2	GPIO	RXP	TXD [3]	RxEN
3	GPIO	CLK	TXD [2]	RxCLK
4	GPIO	TXME	TXD [1]	CRS/ TxBUSY
5	GPIO	TXM	TXD [0]	TxCLK
6	GPIO	TXP	TX_EN	TxD
7	GPIO	TXPE	TX_CLK	TxEN

Table 2-9 Port F Signal Map

Port Bit # PF[*n*]	Func 0 (GPIO)	Func 1 (SerDes)	Func 2 (1/2 MII)
0	GPIO	RXD	TX_ER
1	GPIO	RXM	RX_ER
2	GPIO	RXP	RX_CLK
3	GPIO	CLK	RX_DV
4	GPIO	TXME	RXD [0]
5	GPIO	TXM	RXD [1]
6	GPIO	TXP	RXD [2]
7	GPIO	TXPE	RXD [3]

Table 2-10 Port G Signal Map

Port Bit # PG[*n*]	Function 0 (GPIO)
31:0	GPIO

Table 2-11 Port H Signal Map

Pin PH[*n*]	Func 0 (GPIO)	Func 1 (MII)	Func 2 (CLK)
0	GPIO	CRS	GPIO
1	GPIO	COL	GPIO
2	GPIO	TXD [3]	GPIO
3	GPIO	TXD [2]	GPIO
4	GPIO	TXD [1]	GPIO
5	GPIO	TXD [0]	GPIO
6	GPIO	TX_EN	GPIO
7	GPIO	TX_CLK	GPIO
8	GPIO	TX_ER	GPIO
9	GPIO	RX_ER	GPIO
10	GPIO	RX_CLK	GPIO
11	GPIO	RX_DV	GPIO
12	GPIO	RXD [0]	GPIO
13	GPIO	RXD [1]	CLK_OUT
14	GPIO	RXD [2]	GPIO
15	GPIO	RXD [3]	GPIO

3.0 System Architecture

The central feature of the IP3023 architecture is hardware multithreading, with zero-overhead context switching between hardware threads. All registers that contain context-specific information are duplicated for each of eight hardware threads. The CPU hardware is capable of switching from one hardware thread to another, on a cycle-by-cycle basis with no switching delay. This design enables deterministic and extremely efficient interrupt response, which in turn supports the creation of *software peripherals*. A *software peripheral* is a combination of simple peripheral I/O hardware, and control logic implemented in software, rather than custom peripheral hardware.

3.1 CPU Registers

The IP3023 features 16 general-purpose 32-bit data registers, eight 32-bit address registers (A0-A6, A7/SP), multiply/multiply-accumulate (MAC) output registers, and various other registers. These registers reside in the register address space, an address space separate from both the instruction and data memories. Instructions reference the registers within the register address space directly (as opposed to indirectly through offsets from an address base register). There is no capability for indirect referencing of registers in the register address space.

Every register in the register address space is 32 bits wide.

There are two distinct groups of registers in the register address space:

- Per-Thread Registers
- Global Registers

Some registers are described as read-only. Do not write to a read-only register. Writes to these registers do not change the state of the register, but may cause undesirable side effects.

Some registers are described as write-only. Reads of these registers return undefined results.

3.1.1 Per-Thread Registers

Per-thread registers define the architectural state of one hardware thread. The first 64 registers are per-thread; that is, to support immediate context switching (without the overhead of saving and restoring these registers in software), the per-thread register set is duplicated for each of the eight hardware-supported threads, as shown in Figure 3-1. Table 3-1 shows the locations of these registers in the register space.

Refer also to Section 6.2 for detailed register descriptions.

Table 3-1 Per-Thread Register Map

Address	Register(s)	Description
000-03C	D0–D15	General-purpose data registers.
040-07C	Reserved	
080-098	A0–A6	32-bit address registers.
09C	A7 or SP	32-bit stack pointer, also referred to as A7.
0A0	MAC_HI	Multiply-accumulate result, high 32-bits.
0A4	MAC_LO	Multiply-accumulate result, low 32-bits.
0A8	MAC_RC16	Multiply-accumulate result, rounded and clipped.
0AC	SOURCE3	Implicit third source operand for certain instructions.
0B0	INST_CNT	Count of executed instructions.
0B4	CSR	Condition codes and status register.
0B8	ROSR	Read-only status register.
0BC	IREAD_DATA	IREAD instruction output
0C0	INT_MASK0	Thread interrupt mask.
0C4	INT_MASK1	Thread interrupt mask.
0C8-0CC	Reserved	
0D0	PC	32-bit Program Counter.
0D4-0FC	Reserved	

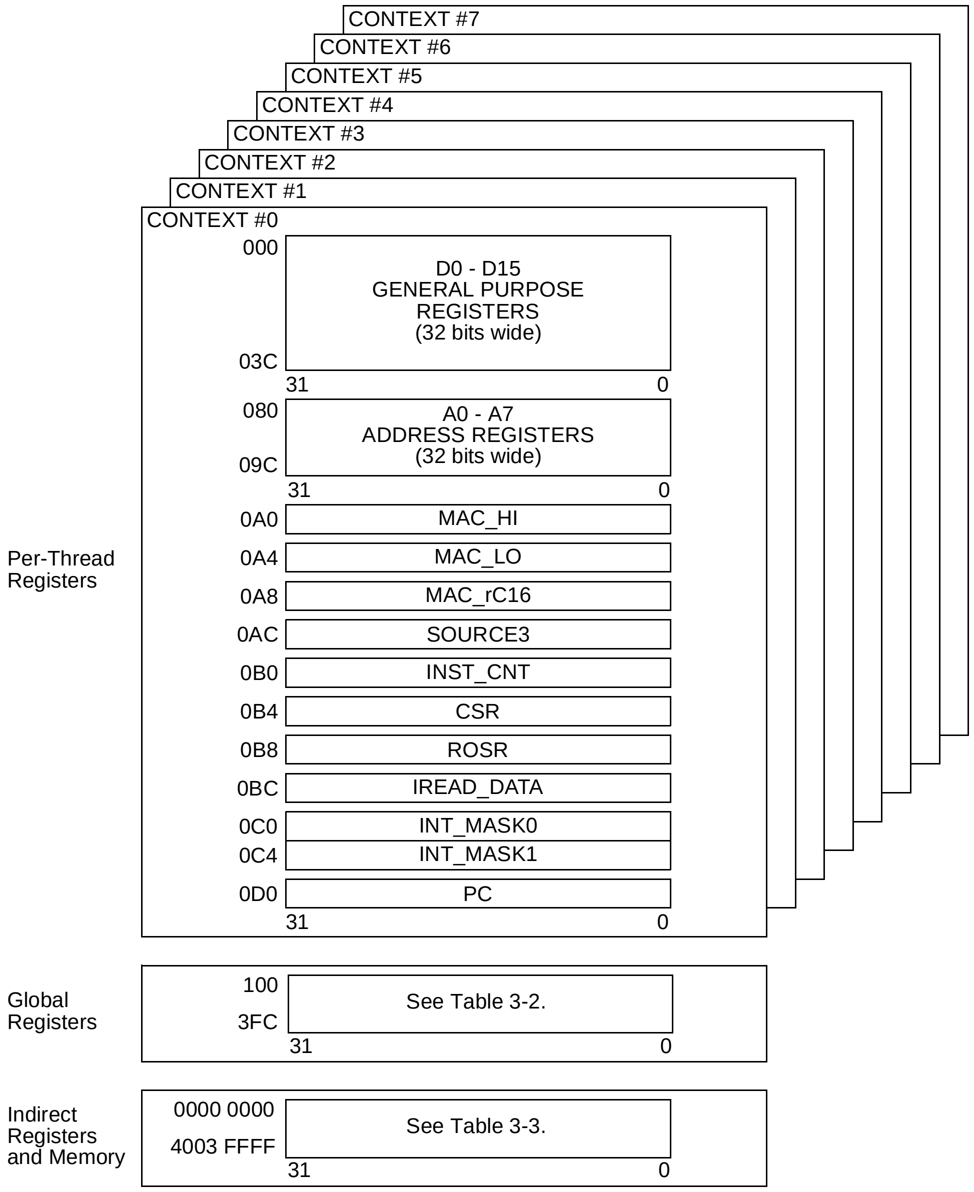


Figure 3-1 Per-Thread, Global, and Indirect Registers, and Indirect Memory

3.1.2 Global Registers

Registers at addresses 0x100 and greater are global; that is, shared among all threads. Table 3-2 shows the addresses of these registers in the register space. Refer also to Section 6.3 for detailed register descriptions.

Registers containing bits that can be set by hardware are generally read-only. To enable software to set or clear bits in these registers, there are associated "write-only" set and clear registers. A value written to a "set" register is atomically OR-ed, on the next cycle, with the corresponding hardware register. The complement of a value written to a "clear" register is atomically AND-ed, on the next cycle, with the corresponding hardware register.

Table 3-2 Global Register Map

Address	Register(s)	Description	Type
100	CHIP_ID	Chip ID.	Read Only
104 108	INT_STAT0 INT_STAT1	Interrupt Status	Read Only
10C-110	Reserved		
114 118	INT_SET0 INT_SET1	Set Interrupt Status	Write Only
11C-120	Reserved		
124 128	INT_CLR0 INT_CLR1	Clear Interrupt Status	Write Only
12C-130	Reserved		
134	GLOBAL_CTRL	Processor function control bits	Read/Write
138	MT_ACTIVE	Threads' active/inactive status	Read Only
13C	MT_ACTIVE_SET	Set bits of MT_ACTIVE register	Write Only
140	MT_ACTIVE_CLR	Clear bits of MT_ACTIVE register	Write Only
144	MT_DBG_ACTIVE	Threads' Debug Active status.	Read Only
148	MT_DBG_ACTIVE_SET	Set bits of MT_DBG_ACTIVE register	Write Only
14C	MT_EN	Multithreading Enable	Read/Write
150	MT_HPRI	Multithreading High Priority Thread mask for non-real-time (NRT) threads	Read/Write
154	MT_HRT	Multithreading Hard Real Time Thread (HRT) mask	Read/Write
158	MT_BREAK	Multithreading BKPT executed mask	Read Only
15C	MT_BREAK_CLR	Clear bit of MT_BREAK register	Write Only
160	MT_SINGLE_STEP	Multithreading Single Step mask	Read/Write
164	MT_MIN_DELAY_EN	Multithreading Minimum Delay Enable mask	Read/Write
16C	PERR_ADDR	Address of a reported memory parity error	Read/Write
170	DCAPT	Data Capture Address	Read/Write
174	DCAPT_PC	Program Counter corresponding to DCAPT	Read Only
178	DCAPT_TNUM	Thread ID and cause corresponding to DCAPT	Read Only
17C	MT_DBG_ACTIVE_CLR	Clear bits of MT_DBG_ACTIVE register	Write Only
180	SCRATCHPAD0	Four scratchpad registers	Read/Write
184	SCRATCHPAD1		
188	SCRATCHPAD2		
18C	SCRATCHPAD3		
190-3FC	Reserved		

3.2 Addressing Model

The IP3023 has separate on-chip data and instruction (program) memories. Using separate data and address buses for the data and instruction memories, instruction fetches and data operand accesses are done concurrently without any contention or waiting. The data memory is dual ported, to allow up to one 32-bit operand read and one 32-bit operand write in each clock cycle. Instruction memory and data memory have nonoverlapping addresses; so, the data and instruction address space can be treated as a single unified 32-bit space. Table 3-3 shows how this space is allocated.

CPU registers belong to an address space separate from data and program memory address spaces.

All memories use byte addressing, although all accesses to instruction memory and registers are in 32-bit word multiples, 32-bit word-aligned. Data accesses vary in width, depending on instruction. Operand addressing in data memory is big-endian – i.e., the most significant byte has the lowest address. Bit numbering within registers and instruction and data memory is little-endian, with bit 0 being the least significant bit.

Table 3-3 Indirect Registers, Data Memory, and Instruction Memory Map

Address Range	Function
Data Space	
0000 0000–0000 07FF	Reserved
0000 0800–0000 083F	HRT Table 0 See Table 6-4.
0000 0840–0000 08FF	Reserved for HRT Table 0 expansion
0000 0900–0000 093F	HRT Table 1 See Table 6-4.
0000 0940–0000 09FF	Reserved for HRT Table 1 expansion
0000 0A00–0000 0AFF	Timers - see Table 6-5.
0000 0B00–0000 0BFF	Debug Mailboxes See Table 5-36.
0000 0C00–0000 0FFF	Reserved
0000 1000–0000 107F	I/O Port A - see Section 6.7.
0000 1080–0000 10FF	I/O Port B - see Section 6.8.
0000 1100–0000 117F	I/O Port C - see Section 6.9.
0000 1180–0000 11FF	I/O Port D - see Section 6.10.
0000 1200 – 0000 127F	I/O Port E See Section 6.11.
0000 1280 – 0000 12FF	I/O Port F See Section 6.12.
0000 1300 – 0000 137F	I/O Port G see Section 6.13.
0000 1380 – 0000 13FF	I/O Port H See Section 6.14.
0000 1400 – 0000 1FFF	I/O port reserved space
0000 2000 – 000F FFFF	Reserved
0010 0000 – 0010 FFFF	On-chip data SRAM (64 KB)
0011 0000 – 003F FFFF	Reserved
0040 0000 – 004F FFFF	Reserved
0050 0000 – FFFF FFFF	Reserved
Instruction Space	
2000 0000 – 203F FFFF (0000 0000 – 003F FFFF*)	Off-chip flash (4 MB)
4000 0000 – 4003 FFFF	On-chip program SRAM (256 KB)

* Flash memory is aliased to several different ranges of addresses. Ubicom software uses the range 2000 0000 – 203F FFFF. At start-up, the CPU begins executing at address 0000 0000. Therefore, start-up code should branch to a location in the 2000 0000 range.

3.3 Instruction Model

Instructions perform memory-memory operations, as well as memory-register, register-memory, and register-register operations. A variety of addressing modes are available. Instructions are 32-bits wide, and execute at the rate of one per cycle.

3.4 Fast Context Switch For Multithreading

A context is all of the state information for a given task or thread – all the information must be saved when the flow of program execution for a given thread is interrupted, so that the thread can restart as if no interrupt had occurred. The context consists of the following pieces:

- Per-thread registers.
- Data memory area used by the given thread.
- Control and status registers of peripheral support logic that is used by the given thread. For example, if Port A is used by a thread, then its setting and status are part of the context.

The IP3023 and its programming environment support fast context switching in the following ways:

- Per-thread register file with 8 sets of the context-dependent registers, one set of registers for each thread.
- Indexed addressing for data memory. The index registers are themselves part of the per-thread register file.
- Compilation, linking, loading tool chain that provides unique base addresses for each task. All addressing modes for the data memory are address-register-based, so it is possible to have multiple instances of a software thread executing on different data sets.

With this hardware support for context switching, each virtual peripheral has a unique view of memory and the programming model that is unaffected by and mostly unaware of other virtual peripherals that may exist.

Furthermore, because the important registers are duplicated for each context, there is no need to save or restore any registers when switching between different threads. Therefore, a context switch can occur in zero-time between instructions.

3.5 Instruction Level Multithreading

Each set of per-thread registers defines a thread. Each thread is identified by a integer in the range 0–7 which corresponds to its entry in per-thread register file.

Several global registers contain information that the CPU uses to schedule execution of the threads:

- MT_ACTIVE, and the corresponding MT_ACTIVE_SET, and MT_ACTIVE_CLR
- MT_DBG_ACTIVE, and the corresponding MT_DBG_ACTIVE_SET, and MT_DBG_ACTIVE_CLR
- MT_EN
- MT_HPRI
- MT_HRT
- MT_BREAK
- MT_BREAK_CLR
- MT_SINGLE_STEP
- MT_MIN_DELAY_EN

All of the above registers are structured as bit maps, where each bit position corresponds to a thread; for example, bit 0 corresponds to thread 0, bit 1 corresponds to thread 1, etc. Bits 31:8 are reserved.

3.5.1 Scheduling Table (HRT)

The IP3023 uses two Hard-Real-Time (HRT) tables to control thread scheduling. The HRT tables are located at fixed memory addresses (shown in Table 3-4). One of the two HRT tables is active and being used by the CPU; the other is available for updates. The HRT Table Select bit in the GLOBAL_CTRL register determines which is the active table.

Each of the 64 HRT table entries is 8 bits wide. Table entries are contiguous in memory, one entry per byte. An HRT entry has the format shown in Table 3-4.

Table 3-4 HRT Entry

Bit Field	Description	
7	End of Table.	
	1 =	The next entry executed will be entry zero of the table indicated by the HRT Table Select bit in the GLOBAL_CTRL register.
	0 =	Not end of table.
6	Unoccupied Entry.	
	1 =	This time slot is available for a Non-Real-Time (NRT) thread.
	0 =	The thread indicated by Thread Number should be scheduled if it is schedulable.
5:3	Reserved.	
2:0	Thread Number.	

Each entry in the table represents an available instruction cycle and specifies the thread (if any) to which that cycle is allocated. Software controls how many of the 64 table entries are actually used by setting bit 7 in the last used entry. The HRT table must have at least one element with bit 7 set.

At each cycle, the CPU steps to the next entry in the current HRT table and determines which thread to execute based on the information in that entry. After it has processed the last entry of the table, it checks the global

HRT Table Select bit and goes to entry zero of the currently selected HRT table.

At power-up or reset, bit 6 is cleared and bit 7 is set in all entries (each entry is an unoccupied end-of-table entry). Software must ensure that the active HRT table has at least one entry with the End of Table (bit 7) set. If no entry has bit 7 set, the result is undefined.

3.5.2 Scheduling Policies

For scheduling purposes, threads are defined as:

- HRT – A thread whose bit in the MT_HRT register is 1. An HRT thread can only be scheduled in time slots allocated to it by the current HRT Table.
- NRT – A thread whose bit in the MT_HRT register is 0. NRT threads can be scheduled both in the HRT table and by the round-robin scheduler.

The IP3023 implements three scheduling policies:

1. Hard-Real-Time (Time-Division Multiplexing) – An HRT thread is guaranteed to receive CPU cycles in proportion to the number of its HRT Table entries.
2. Round-Robin – NRT threads are scheduled on a round-robin basis (in rotation) during the CPU cycles when either no HRT thread is allocated or the allocated HRT thread is not ready.
3. Priority – Among the NRT threads, those threads whose bit in the MT_HPRI register is set to 1 have high priority; others have low priority. No low-priority NRT threads receive a CPU allocation as long as there are active high-priority NRT threads.

Note that it is possible for an NRT thread to have time slots allocated to it in the HRT table. Such a thread participates in round-robin scheduling but is also guaranteed to receive a minimum level of service from the CPU in proportion to the number of its entries in the HRT Table.

3.5.3 Schedulable Threads

Each thread has 3 bits in global bit-mapped registers that determine if it is schedulable:

- MT_EN – This bit is only set or cleared under software control and provides a master thread enable.
- MT_DBG_ACTIVE – This bit is cleared when a BKPT instruction is executed or when a thread performs a single-step. It can also be set or cleared under software control.
- MT_ACTIVE – This bit is cleared when a suspend instruction is executed, and set when the thread receives an interrupt. It can also be set or cleared under software control.

A thread is schedulable when all of these bits are set. If a thread is not schedulable, the hardware thread scheduler will not execute an instruction from that thread.

3.5.4 Hard Real-Time (HRT) Scheduling

The static schedule for HRT threads is specified by the HRT Table.

Figure 3-2 shows an HRT example with three threads in a table that is eight entries long. Thread 1 is scheduled 50% of the time, thread 2 is scheduled 25% of the time and thread 3 is scheduled 12.5% of the time. With the IP3023 clocked at 250 MIPS, this would equate to 125, 62.5, and 31.25 MIPS, respectively. The vacant slot in the last entry of the table guarantees that at least 31.25 MIPS remain available for NRT thread execution.

Each HRT thread is guaranteed to be allocated the instruction slots specified in the table, when it is ready to use them, provided it is schedulable. Thus each HRT thread has guaranteed deterministic performance.

The interrupt latency for each HRT thread is deterministic within the resolution of its static allocation. The pipeline length determines the latency and the time until the thread is next scheduled. The added scheduling jitter can be considered to be the same as an asynchronous interrupt synchronizing with a synchronous clock. For example, a thread with 25% allocation will have deterministic interrupt latency with respect to a clock running at 25% of the system clock.

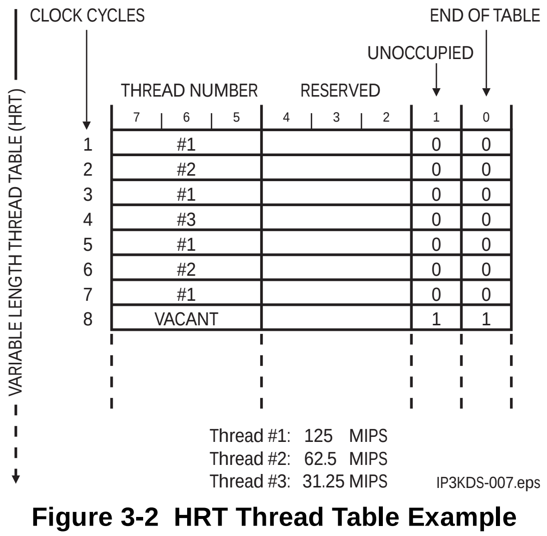


Figure 3-2 HRT Thread Table Example

Although the HRT Table reserves the instruction slots for the hard real-time threads this does not mean that other threads cannot sometimes execute in that instruction slot. For example a UART in thread 3 will actually be idle most of the time. It only needs deterministic performance when it is sending or receiving, and there is no need for it to be scheduled when it is not active. All vacant instruction slots and all slots that are allocated to threads that are not schedulable are used by the scheduler for dynamically schedulable (round-robin) threads.

3.5.5 Round-Robin (NRT) Scheduling

As the name suggests, round-robin threads are scheduled in turn, with one instruction initiated from each schedulable thread. Round-robin threads are scheduled in the vacant slots in the HRT table and in slots where the HRT or NRT thread specified by the table is not schedulable.

Two levels of priority are supported for NRT threads: low and high. Priority is controlled by the thread’s bit in the global MT_HPRI register. If any high priority thread has its MT_EN and MT_ACTIVE bits set (regardless of its MT_DBG_ACTIVE bit), no low priority thread will be scheduled. This is true even if the high priority thread is not ready to execute.

3.5.6 Suspend

A thread can temporarily remove itself from scheduling activity with the SUSPEND instruction. SUSPEND clears the MT_ACTIVE bit for the current thread, so that the thread will not be scheduled. An interrupt condition for that thread asserts the MT_ACTIVE and re-enables normal scheduling of the thread.

3.5.7 Startup

At startup or after reset, thread 0 is active, debug active, and enabled, and all other threads are disabled. Thread 0 begins execution at the fixed flash ROM address 0x0000 0000, and is responsible for initialization; for example:

- Branch to an address in the normal flash address range beginning at 0x2000 0000.
- Load the instruction SRAM;
- Load an HRT table and all thread control registers (including PC);
- Initialize global semaphores and shared memory.

When the initialization is complete, thread 0 enables the other initialized threads, which then are free to execute.

3.6 Programming and Debugging Support

The IP3023 has advanced in-system programming and debug support on-chip. This unobtrusive capability is provided through a dedicated Debug Interface. There is no need for a bond-out chip for software development. This eliminates concerns about differences in electrical characteristics between a bond-out chip and the actual chip used in the target application. Designers can test and revise code on the same part used in the actual application.

Ubicom provides the complete Red Hat GNUPro tools, including C compiler, assembler, linker, utilities, and GNU debugger. In addition, Ubicom offers an integrated graphical development environment which includes an editor, project manager, graphical user interface for the GNU debugger, device programmer, and ipModule™ configuration tool, and profiler.

3.7 Debugging Features

The IP3023 has a number of mechanisms that are intended for use by Ubicom’s debug kernel, and that support an off-chip debugging system. These mechanisms include:

- MT_DBG_ACTIVE active register
- MT_SINGLE_STEP register
- Breakpoint instruction BKPT and breakpoint interrupt
- SUSPEND instruction
- MT_BREAK Register
- Debug mailboxes and the Debug Mailbox Interrupt
- DCAPT register and DCAPT Interrupt
- Parity generation, force value, parity error reset
- Minimum Instruction Delay

3.7.1 Single-Step

The MT_SINGLE_STEP register bit allows a controlling thread to single-step threads that are being debugged. This feature is enabled (on a per-thread basis) by setting the MT_SINGLE_STEP bit that corresponds to the thread being single-stepped. When this bit is set, the thread being debugged is executed as scheduled by the multithreading features. Simultaneously, the CPU clears that thread's MT_DBG_ACTIVE bit, so that the thread will not be activated until software sets the MT_DBG_ACTIVE bit again.

3.7.2 Breakpoints

Debugging breakpoints are supported by the Program Breakpoint interrupt, the BKPT instruction, and the MT_DBG_ACTIVE register.

The BKPT instruction suspends the thread that executes it and clears its MT_DBG_ACTIVE bit. In addition, the BKPT instruction can suspend additional threads and clear their MT_DBG_ACTIVE bits. The source operand is a bit mask that specifies which additional contexts to suspend.

The BKPT instruction asserts the MT_BREAK bit of the current thread (so that the debug kernel knows which thread executed the BKPT instruction) and asserts the Program Breakpoint interrupt (if enabled).

3.7.3 Debug Watchpoint (DCAPT)

The DCAPT register can be loaded with a register, program, or data address. Any write to the address in the DCAPT register triggers the DCAPT Interrupt. The DCAPT register can't be disabled, but can be loaded with a value that can never match. A value that can never match is one with bit 0 equal to 1 and bits 31:13 not equal to all 0's; for example: 0x8000 0001.

The DCAPT Interrupt is also asserted upon detection of out-of-range or operand misalignment errors. Valid address ranges are defined by Table 3-5; all others cause a DCAPT Interrupt.

Whenever the DCAPT Interrupt is asserted, the DCAPT_PC and DCAPT_TNUM registers capture the program counter and thread number of the instruction and the reason for the interrupt .

Table 3-5 Valid Interrupt Addresses

Space	Address Range				Target	Size in Bytes
Register	xxxxxxxx	xxxxxxxx	xxxxxx0x	xxxxxxxx	Registers	512
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	xxxxxxxx	Thread registers	256
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	00xxxxxx	Data registers	64
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	100xxxxx	Address registers	32
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	101000xx	MAC_HI	4
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	101001xx	MAC_LO	4
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	101010xx	MAC_RC16	4
Register	xxxxxxxx	xxxxxxxx	xxxxxx00	101110xx	ROSR	4
Data	00000000	00000000	0000100x	xxxxxxxx	HRT tables	512
Data	00000000	00000000	00001010	xxxxxxxx	Timers	256
Data	00000000	00000000	00001011	xxxxxxxx	Debug mailboxes	256
Data	00000000	00000000	0001xxxx	xxxxxxxx	I/O ports	4096
Data	00000000	00010000	xxxxxxxx	xxxxxxxx	On-chip data SRAM	64K

3.7.4 Debug Mailboxes

The Debug Mailboxes are the software visible portion of the debug port. The debug kernel uses the Debug Mailboxes to receive requests from an external debugging system and to return results. The Debug Mailbox Interrupt signals to the debug kernel regarding arrival or departure of mailbox messages. Refer also to Section 5.10 for more information about the Debug Port and the Debug Mailboxes.

3.7.5 Execution Control

Some of the CPU's instruction execution parameters can be modified by Ubicom's debug kernel to aid in debugging.

When thread's bit in the MT_MIN_DELAY_EN register is set, the Minimum Instruction Delay value of the GLOBAL_CTRL register (refer to Section 6.3.4) is applied. This value is the minimum number of clocks between instructions of the same thread.

3.7.6 Parity Error Control

The Parity Error Interrupt signals memory parity errors to software. When this interrupt is asserted, the PERR_ADDR register contains the address of the error.

In addition, the GLOBAL_CTRL (refer to Section 6.3.4) register controls the following options:

- PGEN_FORCE (Parity Force Value) – The value written to the on-chip instruction memory parity bit when PGEN_EN = 0.
- PGEN_EN (Parity Generation Enable) – A value of 1 enables generation of even parity for all data written to the on-chip instruction memory. This is the normal mode of operation. A value of 0 disables parity generation. This mode is intended for diagnostic purposes only. In this mode, the value of PGEN_FORCE is written to the parity bit for all writes to the on-chip instruction memory.
- PERR_EN (Parity Error Reset Enable) – A value of 1 enables the feature which automatically resets the chip when a parity error is detected, either during instruction access to the on-chip program memory or during data access via `iread` or `iwrite` instructions.
- A value of 0 disables the feature. Note that the Parity Error Interrupt function is not masked by this bit.

3.8 Interrupts and Exceptions

Interrupts are signaled by setting bits in the global Interrupt Status register. This is a 64-bit register (in two 32-bit parts, INT_STAT0 and INT_STAT1), with each bit representing a potential interrupt source. When a bit is set to 1, it asserts the associated interrupt condition.

Bits in the Interrupt Status register corresponding to I/O interrupts give the state of the corresponding I/O interrupt(s). Other bits can be set by hardware or by software. If there is no hardware source associated with a particular bit, that bit represents a software interrupt. However, even if there is a hardware source (other than I/O) associated with an interrupt status bit, the bit can still be set by software. This makes it possible to simulate interrupts for software testing.

Once an interrupt status bit is set, it remains set until explicitly cleared by software. No automatic interrupt acknowledge signal is sent to an originating peripheral device – neither when the interrupt status bit is set, nor when an interrupt handler responds to it. If an acknowledgement is needed, it is the responsibility of the interrupt handling software to send it, by writing to the appropriate peripheral register.

Note: I/O interrupts must be cleared by writing to the appropriate I/O control register, not by directly clearing the INT_STAT bit.

Section 6.3.2 and Section 6.3.3 show the mapping of interrupt status bits to specific interrupt sources.

3.8.1 INT_STAT0 Register

The INT_STAT0 register is dedicated to interrupts from software and timers.

A block of eight fine-grained timer interrupts is associated with a corresponding block of eight 32-bit timer registers. When the value held in a given timer register matches the value of the global cycle count register, the corresponding interrupt is asserted. Refer to Section 5.9 for details about timers.

3.8.2 INT_STAT1 Register

INT_STAT1 is dedicated to internal interrupts and I/O interrupts

- Program Breakpoint. This interrupt indicates that one or more threads has been halted, as a result of execution of the BKPT instruction. Refer to Section 3.7 for more information about debugging features.
- DCAPT Interrupt. Indicates that debug event has occurred. Any of the following events can assert this interrupt:
  - A write to data or code memory matches the address specified in the DCAPT register.
  - A data memory access in an illegal memory range. Valid address ranges are defined by Table 3-5; all others cause a DCAPT Interrupt.
  - A 2- or 4-byte wide data memory access (source or destination) has incorrect alignment.

  Whenever this bit is set, the DCAPT_PC and DCAPT_TNUM registers capture the PC and thread number of the instruction and the reason for the interrupt . New events replace the information saved from earlier events. A DCAPT interrupt does not suspend the thread that caused it.
- Debug Mailbox Interrupt. Indicates that a debug message is waiting or has been successfully sent.
- Real-Time Compare Register Interrupt. Refer to Section 5.9 for details about the real-time compare register.
- Memory Parity Error Interrupt. A memory parity error occurred. The PERR_ADDR register contains the address of the error.
- I/O Interrupts. Most I/O functions can generate a variety of interrupts, which are visible in INT_STAT1. Each I/O function has a Interrupt Status register giving the cause of the interrupt and the interrupt bit itself. Several I/O interrupts might be shared by a single INT_STAT bit.

3.8.3 Thread Interrupt Mask

Each hardware context has a 64-bit Interrupt Mask register, that determines the interrupts to which it responds. The mask is logically ANDed with the contents of the Interrupt Status register; if the result is non-zero, an interrupt condition is signaled to the associated hardware thread, setting the INTERRUPT CONDITION bit of its ROSR register. If the thread is currently suspended, it is made active. If it is currently active, it remains active, continuing normal execution. However, if it executes a SUSPEND instruction, the presence of the pending interrupt will immediately reactivate it.

The Interrupt Mask register is a per-context read-write register. It is normally written only at start-up, however, to configure the assignment of interrupts to hardware threads.

3.8.4 Multithreading Break

The MT_BREAK global read-only register is closely associated with the Program Break interrupt. It has one bit per hardware thread. If the bit for a given thread is set, it indicates that the thread is halted for a break condition. The interrupt handler for the Program Break interrupt can read this register to determine which thread is halted for a break condition.

Bits in the MT_BREAK register are cleared by software, by writing to the Multithreading Break Clear (MT_BREAK_CLR) register. Clearing one of these bits does not restart the corresponding thread; setting the MT_DEBUG_ACTIVE bit accomplishes that.

3.8.5 Forcing an Interrupt

As mentioned in connection with the Interrupt Mask register, the presence of an interrupt condition signaled to a thread serves merely to reawaken the thread, if it is suspended, or to cancel its suspension, if it is running and executes a SUSPEND instruction. For high priority interrupts with dedicated handler threads, the system design requirement for the interrupt handling time to be less than the inter-arrival time of the interrupt guarantees that the handler will be suspended when the interrupt arrives. Interrupt response, in that case, is immediate.

When independent interrupts share a common interrupt handler thread, it is possible for the handler to be active, responding to a previous interrupt, when a new interrupt arrives. Handling of the new interrupt will then be delayed until handling of the previous interrupt is completed, and the interrupt handler thread issues a SUSPEND.

In order to minimize interrupt latency for interrupts handled by a common handling thread, the handling functions should be kept short. In some cases, that means using the common interrupt handling thread as a "front end", to force a vectored interrupt to an extended ISR in a thread running a lower priority background process. The instruction sequence to accomplish this is:

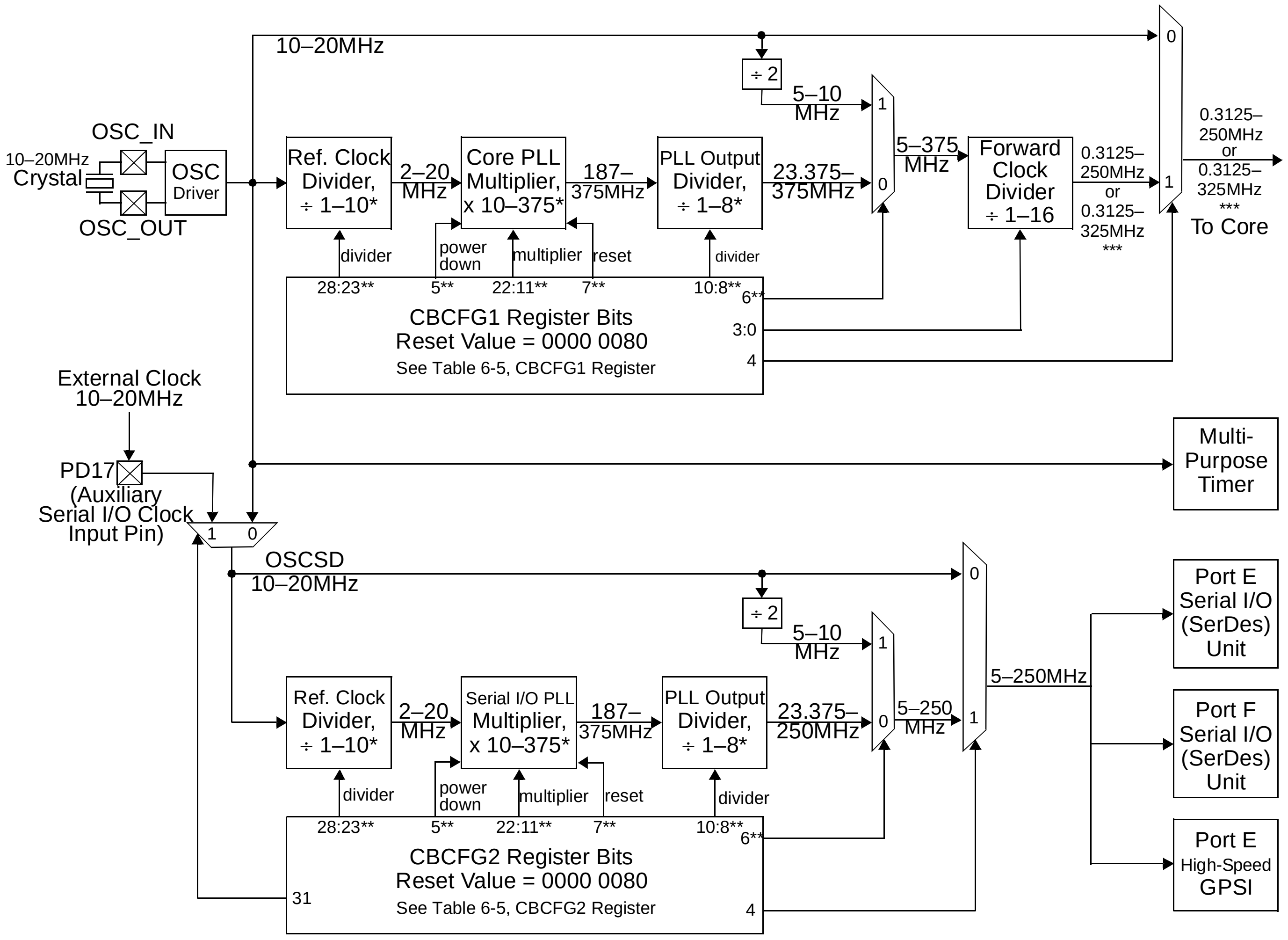
- Halt the target thread by clearing its bit in the Multithreading Enable (MT_EN) register. Note that this does not force the cancellation of any instructions for that thread that are already in the pipe; it merely keeps the thread scheduler from allocating any more cycles to the target thread, until it is re-enabled;
- Wait until all instructions for that thread have cleared the pipeline;
- After setting the source thread select field in the CSR to the target thread number, copy its PC and CSR values to control memory, where they can be accessed later.
- After setting the destination thread select field in the CSR to the target thread number, write the desired ISR address and appropriate CSR value to its PC and CSR;
- Re-enable the thread by setting its bit in the Multithreading Enable register.
- Use the SETCSR instruction to recover the control thread's own destination context.

3.9 Clock Circuitry

Figure 3-3 shows the logic for producing core and I/O clocks. Note that the Forward Clock Divider's values can be changed on the fly without adverse effects. The clock registers location and layout are specified in Section 6.5.

Refer to the Programmers Reference Manual for detailed programming instructions.

An external crystal must be connected between OSC_IN and OSC_OUT, as discussed in Section 3.10. External clocks into OSC_IN are not supported.



* Capability exists to program to any integer value in this range, but, for reliable operation, a value must be used which results in the output frequencies shown in this figure. **Bit field values of N cause divide or multiply by N+1.** For instance if CBCFG1 Register has bits 28:23 = 000000, then the Reference Clock Divider will divide by 1.

** CBCFG1 and CBCFG2 bits 28 through 5 must not be changed while CBCFG1 bit 4 = 1 (this will cause clock glitches). The Forward Clock Divider's values can be changed on the fly without adverse effects. To reduce the core to the minimum possible frequency (OSC_IN frequency divided by 32), clear bit 4, then set bits 6 and 3:0 (and set bits 5 and 7 if the lowest power is desired), then set bit 4.

*** Only the 325MHz version is rated for 325MHz. The 250MHz version is rated for a max of 250MHz.

Figure 3-3 Clock Logic

3.10 Crystal Oscillator

Figure 3-4 shows the connections for attaching a crystal to the OSC oscillator. The crystal is connected across the OSC_IN and OSC_OUT pins. There is about 4pf of capacitance on each of OSC_IN and OSC_OUT pins to DVss. A parallel resonant crystal is recommended that has a maximum ESR of 30 ohms at 20MHz, 60 ohms at 10MHz. A feedback resistor (Rf) of 150K ohms must be connected between OSC_IN and OSC_OUT for reliable crystal startup.

The crystal manufacturer's load capacitance rating (CL) should be equal to (C1 x C2) / (C1 + C2), where C1 = capacitance on OSC_IN (4pF + stray board capacitance + added capacitance), and C2 = capacitance on OSC_OUT (4pF + stray board capacitance + added capacitance). The trace length between the OSC pins and the crystal should be as short as possible, to avoid noise coupling.

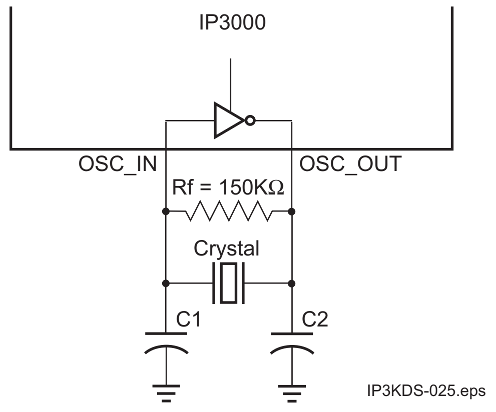


Figure 3-4 Crystal Connection

3.11 Clock Output Generator

The IP3023 provides two independently programmable clock output (CLK_OUT) signals on pins PB6 and PH13. The CLK_OUT signal is derived from the core clock by a clock divider circuit. The user can program the clock divider by writing to a CLK_DIV[7:0] field. The core clock frequency is divided by N+1, where N is the value in the CLK_DIV[7:0] field. To accomplish the division, an internal counter samples CLK_DIV[7:0], then counts down to 0, then samples CLK_DIV[7:0] again, and so on. The state of the counter (delayed by two core clock cycles) is presented to the user in a PHASE[7:0] status field.

The CLK_OUT signal relates to PHASE[7:0] in the manner shown by the examples in Figure 3-5 (odd divisor) and Figure 3-6 (even divisor). For an odd divisor, CLK_OUT goes high when PHASE[7:0] reaches 0, and goes low when PHASE[7:0] reaches (D+1)/2, where D is the divisor. This produces a slightly asymmetric waveform, as shown. For an even divisor, CLK_OUT goes high when PHASE[7:0] reaches 0, and goes low when PHASE[7:0] reaches D/2. This produces a symmetric waveform. If CLK_DIV[7:0] = 0, the output is constant low.

The value in CLK_DIV[7:0] can be changed at any time, without causing ill behavior, since the counter will always finish counting down to 0 before sampling CLK[7:0] again. The PHASE[7:0] status field allows the user to properly align a software process with the clock divider.

CLK_OUT is available on PB6 when Port B Function 1 is selected. CLK_DIV[7:0] is in bits [7:0] of the Port B SDRAM Function Control 2 register. PHASE[7:0] is in bits [15:8] of the Port B SDRAM Function Status 0 register. If PB6 is used as an SDRAM Clock, CLK_DIV[7:0] should remain at a constant value of 3 (divide by 4).

CLK_OUT is available on PH13 when Port H Function 2 is selected. CLK_DIV[7:0] is in bits [7:0] of the Port H Clock Function Control 2 register. PHASE[7:0] is in bits [7:0] of the Port H Clock Function Status 0 register.

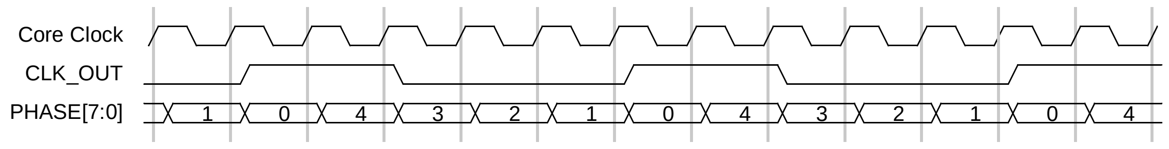


Figure 3-5 Clock Behavior when CLK_DIV[7:0] = 4 (divide by 5)

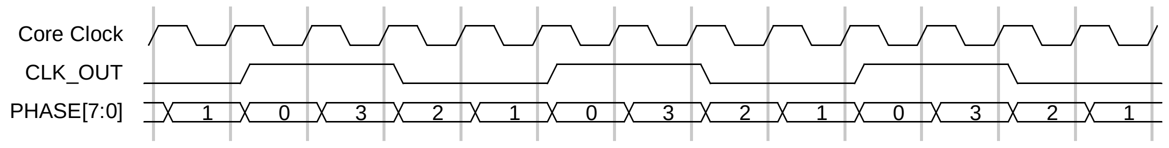


Figure 3-6 Clock Behavior when CLK_DIV[7:0] = 3 (divide by 4)

3.12 Reset

The following sources are capable of causing a chip reset:

- Power-on
- Debug port
- Watchdog timer (when enabled in Multipurpose timer)
- Parity error (when PERR_EN is set to 1)
- External reset ($\overline{\text{RST}}$ pin)

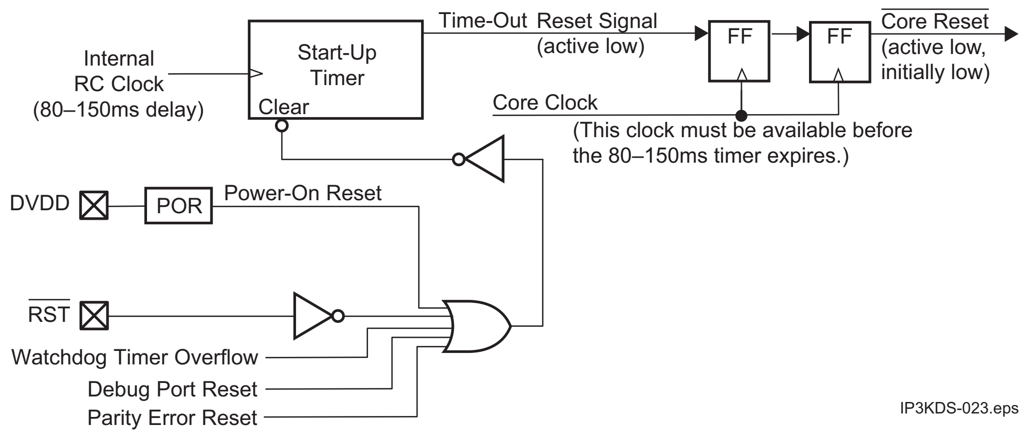
Each of these reset conditions causes the CPU to initialize the HRT and begin executing thread 0 at address 0000 0000.

All Port pins are tri-stated while $\overline{\text{RST}}$ is held low, including Flash port, except see Note 2 in Table 2-3.

The IP3023 incorporates a Power-On Reset (POR) detector that generates an internal reset as DVdd rises during power-up. Figure 3-7 is a block diagram of the reset logic. The startup timer controls the reset time-out delay. The reset latch controls the internal reset signal. On power-up, the reset latch is cleared (CPU held in reset), and the startup timer starts counting once it detects a valid logic high signal on the $\overline{\text{RST}}$ pin, and all other reset sources are deasserted. Once the startup timer reaches the end of the timeout period, the reset latch is cleared, releasing the CPU from reset.

Reset reason flags in the RSTFLAG register (refer to Table 6-5) indicate possible sources of the reset. Separate bits for each reason can be used by software to determine the cause of the reset.

Figure 3-7 On-Chip Reset Circuit Block Diagram



4.0 Instruction Set

4.1 Operand Addressing

The IP3023 has data types of three principle sizes: 8-bit byte, 16-bit short word and 32-bit long word, as shown in Figure 4-1. There is also a 48-bit data type, used only for accumulator results in the MAC register. The byte ordering for operands in memory is big-endian, although bit numbering within registers is little-endian (as shown in Figure 4-1). Big-endian format means that the address of the operand refers to the byte address of the most-significant byte, and bytes are in memory in the order of most to least significant. For example, storing the 16-bit operand 0x1234 at address 0x1000 means that 0x12 is stored at address 0x1000 and 0x34 is stored at address 0x1001.

Both the program and data spaces are byte addressed, and operand addresses must be naturally aligned – that is, they must be integer multiples of the operand size. Except for the IREAD instruction, if an operand address is misaligned, execution does not halt, and some value for the operand is returned, but the value returned in that case is not defined. If the misaligned address is a target, the target may or may not be modified, and additional bytes near the target may be modified. When a reference is made to a misaligned data space operand, as a source or target, the DCAPT interrupt bit is set, and the PC and thread number are captured. Unaligned addresses used by IREAD accesses to program memory are not detected; the low-order two bits are ignored.

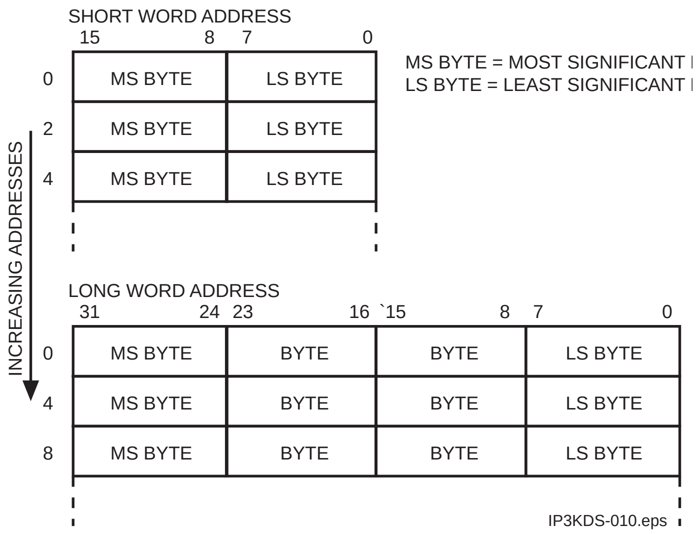


Figure 4-1 Big-Endian Data Formats of IP3023.

4.2 Addressing Modes

Most of the IP3023 instruction formats (as shown in Figure 4-2) use a small number of common fields, which are aligned for ease of decoding. The most important of these are the 11-bit source-1 and 11-bit destination operand specifiers, and the 5-bit source-2 field. The latter may contain either a 5-bit unsigned immediate value, or a data register number (in the right-most 4 bits), depending on the specific instruction. A five-bit immediate value is normally a bit number or a shift count, again depending on the specific instruction.

The 11-bit source-1 and the 11-bit destination fields are defined the same. This 11-bit field is used to select one of the following addressing modes (refer to Table 4-1 for more detail):

- Direct Addressing of the register address space
- Register Indirect with 7-bit unsigned offset
- Register Indirect with pre or post increment or decrement
- Register Indirect with indexing
- Immediate: 8-bit value, sign extended to size of operand type

An immediate value in the destination specifier is unusual, but it can be used to prevent write back of the instruction result value to any real destination. The instruction is then executed 'for side effects only' – i.e., setting the condition codes.

To encode all of the above in only 11-bits, variable length encoding is used. The minimum length encoding of a 1-bit is used for the register indirect with offset mode, allowing the maximum number of bits for offset. This results in the following two formats:

Register Indirect With Offset:	`1 i i A A A i i i i i`
Other Addressing Modes:	`0 x x A A A n n n n n`

...where:

"i" :	Indicates immediate value bits;
"A" :	Indicates the three address bits that select one of the 8 address registers;
" x" and "n" :	Other fields used by other addressing modes.

Details of this encoding scheme are shown in Figure 4-3.

In the Harvard architecture model, addressing modes for data and program memory space have to be considered separately. For data address space, the supported addressing modes are described above. For program space, both register indirect and PC relative addressing modes are supported. Addressing modes for both spaces are summarized in Table 4-1.

4.2.1 The Register Address Space

Register addressing mode is used to address the general-purpose registers, D0-D15, as well as the address registers A0-A7, and all on-chip control registers. It is the only mode that can access the core's control registers, because the address space in which these registers reside is not a subset of the general memory address space. A register in the register addressing space cannot be accessed through a regular memory addressing mode that happens to resolve to the same numerical value.

The register addressing space is 256 registers in length, or 1024 bytes. The byte address specified in an assembler statement is right shifted two bits by the assembler, to generate the 8-bit register address offset. Although the assembler syntax requires a byte address, what is addressed in this mode is not bytes, but a space of 256 32-bit register locations. The register addressing space covers access to the following registers:

- All Programmers' Model registers described in Section 3.1.
- On-chip control and status registers for overall chip and timer control.

Because the register addressing space is absolute, an assembler include file of EQU statements can be used to define symbolic names for all the registers. Then instructions can access these registers directly using their symbolic name as a source or destination operand.

Table 4-1 Addressing Modes

Space	Type	ASM Syntax	Effective Address (EA)
Operand	Register	$xx or Register Mnemonic	No EA. Register address is 10 bits: (8-bit register number) || 00
	Indirect	(An)	EA = An
	Indirect with Offset	offset(An)	EA = An + offset; PDEC only: EA = An - offset (in range 4 to 512); Assembly Syntax: Offset specified in bytes; Opcode Coding: Byte Operand: Offset = 7-bit unsigned immediate value; 16-Bit Operand: Offset = 7-bit unsigned immediate value || 0; 32-Bit Operand: Offset = 7-bit unsigned immediate value || 00; PDEC Operand: Offset = 11111111111111111111111 || 7 bit immediate || 00
	Indirect with Post-Increment	(An)delta++	Step 1: EA = An ; Step 2: An ← An + delta Assembly Syntax: delta specified in bytes; Opcode Coding: Byte Operand: delta = 4-bit signed immediate value; 16-Bit Operand: delta = 4-bit signed immediate value || 0; 32-Bit Operand: delta = 4-bit signed immediate value || 00.
	Indirect with Pre-Increment	delta(An)++	Step 1: An ← An + delta Step 2: EA = An ; Assembly Syntax: delta specified in bytes; Opcode Coding: Byte Operand: delta = 4-bit signed immediate value; 16-Bit Operand: delta = 4-bit signed immediate value || 0; 32-Bit Operand: delta = 4-bit signed immediate value || 00.
	Indirect with Index	(An,Dn)	EA = An + (Dn << $\log_2$(operand size in bytes))
	Immediate	#xxxx #xx	Operand is 16-bit or 8-bit immediate value taken from instruction. Value is sign-extended to 32-bits before use. For 8 bit immediate in general source-1, the EA is the 32-bit sign extended immediate.
CALLI Instruction	Indirect	offset(An)	PEA = An + (sign-extended coded offset <<2) Assembly Syntax: Offset is specified in bytes as a signed 18-bit number. This number is right shifted by two bits for instruction coding. Coded Offset = Assembly Offset[17:0] >> 2
CALL Instruction	Relative	offset(PC)	PEA = PC + (coded offset<<2) Assembly Syntax: Offset is specified in bytes as a signed 26-bit number. This number is right shifted by two bits for instruction coding. Coded Offset = Assembly Offset[25:0] >> 2
JMPcc Instruction	Relative	offset(PC)	PEA = PC + (Opcode Offset)<<2 Assembly Syntax: Offset is specified in bytes as a signed 23-bit number. This number is right shifted by two bits for instruction coding. Opcode Offset = Assembly Offset[23:0] >> 2
Notation: EA: Data Effective Address; PEA: Program Space Effective Address			

4.3 Instruction Set Summary

The instruction set has a fixed-length 32-bit instruction word, and the internal data path is 32 bits wide.

The CSR register (refer to Section 6.2.1 on page 91) contains two sets of condition codes: 16-bit condition codes and 32-bit condition codes. Both sets of condition codes are calculated for the results of arithmetic and logical instructions all the time. The programmer chooses the appropriate set for conditional jump instructions. The bits of both sets of codes have the same symbols, which are defined in Table 4-2.

Table 4-2 16-Bit and 32-Bit Condition Codes

Symbol	Description
N	Negative. Set if result is negative, cleared otherwise.
Z	Zero. Set if result is zero, cleared otherwise. For ADDC and SUBC the Z bit is cleared if the result is nonzero, and left unchanged if the result is zero. This bit does not include any testing of the carry bit.
V	Overflow. For arithmetic operations where overflow is a meaningful possibility, the V bit is set if overflow occurs, and cleared if it does not. For other operations, the bit remains unchanged from its value before the operations.
C	Carry. For arithmetic operations where a carry out is a meaningful possibility (add, subtract, and compare), the C bit is set to the value of the carry out. For subtract and compare operations the C bit contains the complement of the borrow.

The following points apply to all instructions, in the instruction description tables that follow:

- The ".size" field in instruction syntax refers to the data memory width of operands, and not the instruction width or the ALU result width; both the latter are always 32-bits.
- With arithmetic, logical, and shift operations, if the source is 16-bits, it is always sign extended to 32-bits before the operation to match the width of the internal data path. Only MOVE.1 and MOVE.2 zero-extend a smaller source operand, when the destination is a 32-bit register (i.e., any register in the register address space – most frequently a data register). The 16-bit condition codes allow operation with 16-bit unsigned integers in memory.
- All arithmetic and logical operations are performed in 32-bit resolution. If the destination is in memory and the instruction's operand size is only 16 bits, it is the lower 16 bits of the 32-bit result that is written to memory.
- The source-2 operand, for instructions with more than one input operand, is always either a 32-bit data register, or (in the case of shift and bit field instructions) a 5-bit zero-extended immediate value.
- Data registers, address registers, and other registers in the register address space, used as source or destination operands, are always 32-bits wide.
- An immediate 8-bit value in the source-1 operand specifier is always sign-extended to 32-bits before use.
- The tables indicate which condition flags are set by each instruction. In general, any instruction which computes a 32-bit result will set both the N and Z flag bits. Only instructions that can generate a carry or overflow will set the C or V bits. BTST, BSET, and BCLR set only the Z bit. MOVE instructions do not affect the flags.
- The indicated condition flag bits are always set independently in both the 16- and 32-bit condition codes. Thus, if a 32-bit result were all 0's in the lower 16 bits, but had some non-zero bits in the upper 16 bits, the 16-bit condition result would be '1' in the Z bit and '0' in the N bit. The 32-bit result would be '0' in the Z bit and a copy of bit 31 in the N bit.

Example: As the following two instructions demonstrate, the size refers to data memory access size. The arithmetic and logical operations are always performed at 32-bit resolution.

```
; op      dest, s1, s2
ADD.2     (A1),(A0),D2     ; Read 16-Bits →
                           ; Add 32-bits → Store 16-bits
ADD.2     D1,(A0),D2       ; Read 16-Bits →
                           ; Add 32-bits → Store 32-bits
```

4.3.1 Arithmetic and Logical Operations

Integer arithmetic support consists of the basic operations of add, subtract, multiply, and multiply-accumulate (MAC). Logical operations include the four basic operations of AND, OR, XOR, and NOT. They perform bit-wise Boolean operations on operands.

The following paragraphs discuss selected instructions in more detail.

ADDC and SUBC

ADDC and SUBC use the C-bit in the 32-bit condition code to implement extended precision arithmetic operations. The C-bit is used for any carry and borrow between different 32-bit words of an extended operand (there is no ADDC.2 or SUBC.2). For SUBC, the complement of the C-bit on input is the “borrow” value for the operation. The borrow is effectively added to the right-hand operand (the subtrahend) before it is subtracted from the left hand operand. (In practice, what that means is that, whereas normal subtraction is implemented by adding the logical complement of the right hand operand to the left hand operand, with a forced '1' as carry in, SUBC uses the input value of the C-bit as the carry in.)

The Z bits are treated differently for ADDC and SUBC than for other instructions. If the result is nonzero, the Z bit is cleared, but if the result is zero, the Z bit is not changed. When adding multiprecision numbers, first an ADD instruction will set or clear the Z bit for the least significant 32 bits. Subsequent ADDC instructions can only clear the Z bit. After the sequence of ADD and ADDC, the Z bit will be set if the multiprecision result is zero.

There is limited scope for inserting instructions between the instruction that sets the carry flag value and the ADDC or SUBC instruction intended to use it. MOVE, and other instructions that don't affect the C flag are safe, but loop end tests are problematic. A loop end test will normally affect the flag value. Therefore, it is desirable to use in-line expansions, rather than loops, for extended precision arithmetic. Since the normal IP3023 arithmetic operations are 32 bits wide, it only takes a single ADD, followed by one ADDC, to perform a 64-bit extended precision add.

In rare cases, where very long extended precision operations must be implemented, it is possible to use the LEA instruction to decrement a loop counter register, and use an EXT instruction to set the Z and N bits of the condition codes, without affecting the C and V flags. That permits a conditional branch on non-zero for the loop end test, at the cost of the extra EXT.

MULF and MAC

These perform multiply and multiply-accumulate functions on 16-bit fixed point, or so-called “fractional” data types (S.15), most commonly used by Digital Signal Processing algorithms. MULF generates a 32-bit number in the S.31 format, and stores it to the 48 least significant bits of the MAC accumulator register, sign-extending the 32-bit result to 48 bits. The upper 16 bits of MAC_HI are cleared by MAC and MULF. MAC performs an implicit MULF operation, and adds the 32-bit result to the 48-bit accumulator value. The format of the accumulator is S16.31. Source-2 can be a Dn register or a 5-bit immediate representing a number in the range 0 to $31*2^{-15}$.

MULS and MULU

These perform multiply functions on signed or unsigned 16 bit integers. The result is sign extended or zero extended to 48 bits and stored into MAC_HI and MAC_LO. The upper 16 bits of MAC_HI are cleared. Source-2 can be a Dn register or a 5-bit unsigned immediate.

MAC_RC16

The MAC_RC16 register is set by any instruction that implicitly targets MAC_HI and MAC_LO. It is not modified by an instruction that explicitly targets those registers. The result being placed in MAC_HI/MAC_LO is considered to be a number in S16.31 format. This is a two's complement 48 bit fractional number with 31 bits after the decimal point. This number is first rounded to have 15 bits after the decimal point. To round, the value of the 16th bit after the decimal point is added to the 15th bit after the decimal. After rounding, the result is clamped to be in the range 0.111111111111111 (binary) to 1.000000000000000 (binary). The sign of the clamped value is the same as the sign of the original number, before rounding, to prevent the highest positive number from being rounded to the lowest negative number. After rounding and clamping, the result is sign extended to 32 bits and stored in the MAC_RC16 register. The resulting format has 17 copies of the sign bit, a decimal point, and 15 fractional bits, in two's complement.

CRCGEN

This is a special purpose instruction for efficient calculation of CRC values in message protocols, such as Ethernet, and bit-stream scrambling. It models the operation of a Linear Feedback Shift Register (LFSR), for any generating polynomial of order 32 or less. It processes eight bits of input at a time. It is defined as follows:

Syntax: CRCGEN s1, s2

Inputs: s1 – next data byte, B (general source, typically from memory)

s2 – generating polynomial, P (Dn register or 5 bit immediate)

MAC_LO – current CRC value, C

Outputs: MAC_LO – new CRC value, C'

MAC_HI – scrambled output byte, S, shifted into the most significant byte

MAC_RC16 - S16.15 image of MAC-HI, MAC_LO

Operation:

```
X = (C ^ B) & 0xFF;
F = X;
for (i=0; i<8; i++) F = (F>>1)^(F&1 ? P : 0);
S = C & 0xFF;
MAC_LO = F ^ (C >> 8);
MAC_HI = (MAC_HI >> 8) | (S << 24);
MAC_RC16 ← S16.15(MAC_HI, MAC_LO)
```

Table 4-3 Arithmetic and Logical Instructions

Mnemonic & Operands	**Description**	**Flags (for both 16-bit and 32-bit operands)**	**Operand Size (Bytes)**
ADD.2 d,s1,s2 ADD.4 d,s1,s2	*16/32-Bit Add Operation*: d ← s1 + s2	C,Z,N,V	2, 4
ADDC d,s1,s2	*32-Bit Add with Carry*: d ← s1 + s2 + C(32-bit C Flag)	C,Z,N,V	4
AND.[2/4] d,s1,s2	d ← s1 AND s2	N,Z	2, 4
CMPI s1,#imm16	*16-Bit Compare with Immediate Value*: s1 – (sign-extended #16-bits)	C,Z,N,V	2
CRCGEN s1,s2 CRCGEN s1,#POLY	*32-bit incremental CRC Generation Instruction:* MAC_LO holds current CRC; s1 specifies the next input byte s2 specifies the generating polynomial MAC_LO ← CRC(MAC_LO, s1, s2) MAC_HI[23:0]← MAC_HI[31:8] MAC_HI[31:24]← scrambled output byte MAC_RC16 ← S16.15(MAC_HI, MAC_LO)		1
LEA.1 d,s1 LEA.2 d,s1 LEA.4 d,s1	*Load Effective Address*: d ← EA(s1); Calculates the effective-address for the referenced data operand, and stores the address into destination. S1 must not be Register addressing mode. The destination size is always 32 bits. If the destination is an An register (and the Destination Thread Select bit in the CSR is **not** set), a fast path eliminates hazards with a later An use.	-	1, 2, 4 (Source)

Table 4-3 Arithmetic and Logical Instructions

Mnemonic & Operands	Description	Flags (for both 16-bit and 32-bit operands)	Operand Size (Bytes)
PDEC d,s1	d ← EA(s1); Identical to LEA.4, except that the 7-bit offset in base+offset addressing mode is extended to 32 bits by adding 23 1-bits (11111111111111111111111 || 7-bit immediate || 00).		4
MAC s1, s2 MAC s1, #imm	The Dn field is coded as zero 16 x 16-bit signed Fractional Multiply-Accumulate: {MAC_HI[15:0]:MAC_LO} ← {MAC_HI[15:0]:MAC_LO} + SE48((s1 * s2)<<1) MAC_HI[31:16] = 0 MAC_RC16 ← S16.15(MAC_HI:MAC_LO)	-	2
MULF s1, s2 MULF s1, #imm	The Dn field is coded as zero 16 x 16-bit Signed Fractional Multiply: {MAC_HI[15:0]:MAC_LO} ← SE48((s1 * s2)<<1) MAC_HI[31:16] = 0 MAC_RC16 ← S16.15(MAC_HI:MAC_LO)	-	2
MULS s1, s2 MULS s1, #imm	The Dn field is coded as zero 16 x 16-bit Signed integer Multiply: {MAC_HI[15:0]:MAC_LO} ← SE48(s1 * s2) MAC_HI[31:16] = 0 MAC_RC16 ← S16.15(MAC_HI:MAC_LO)	-	2
MULU s1, s2 MULU s1, #imm	The Dn field is coded as zero 16 x 16-bit Unsigned Integer Multiply: {MAC_HI:MAC_LO} ← ZE48(s1 * s2) MAC_HI[31:16] = 0 MAC_RC16 ← S16.15(MAC_HI:MAC_LO)		2
NOT.[2/4] d,s1	d ← NOT s1	N,Z	2, 4
OR.[2/4] d,s1,s2	d ← s1 OR s2	N,Z	2, 4
SUB.2 d,s1,s2 SUB.4 d,s1,s2	*16/32-Bit Subtract Operation*: d ← s1 - s2	C,Z,N,V	2, 4
SUBC d,s1,s2	*32-Bit Subtract with Carry*: d ← s1 – s2 – !C (32-bit C Flag) where: C = 1 if there is no borrow.	C,Z,N,V	4
XOR.[2/4] d,s1,s2	d ← s1 XOR s2	N,Z	2, 4

Notes:·
Fractional multiply has an implicit left-shift by one; otherwise, it would end up with two sign bits.·
SE48: Indicates sign-extension to 48 bits.·
S16.15: The 48-bit accumulator value is rounded, with saturation, to a 16-bit fractional value (S.15 format), then sign-extended to 32 bits (S16.15 format).

4.3.2 Shift and Bit-Field Operations

Table 4-4 Shift and Bit-Field Instructions

Mnemonic & Operands	Description	Flags

ASR.[2/4] d, s1, s2
ASR.[2/4] d, s1, #cnt
	Arithmetic Shift Right.
	shct ← s2, or shct ← #cnt
	d ← s1[31]^{shct} || s1[31:shct]
	If 's1' is a 16-bit operand, it is sign-extended to 32 bits before the operation; 'd' must be a data register.
		N, Z

BFEXTU d, s1, s2 BFEXTU d, s1, #cnt	*Bit Field Extract*. Extracts a bit-field from 32-bit s1 operand and places it into LSBs of destination. s2 (Dn or 0-extended 5-bit immediate) contains the parameters: Bits 4:0: Bit-Field Length; Bits 12:8: Bit-Field Start position (low order bit); The result is zero-extended to 32 bits. The destination must be a data register. If Length == 0, then result = 0; If start + length ≥ 32 → Then, cut at the left edge.	N, Z

BFRVRS d, s1, s2
BFRVRS d, s1, #cnt
	Bit Field Reverse. Reverse bit ordering 32-bit s1 operand, followed by logical right shift by shift count:
	d ← (Reverse(s1)) >> s2[4:0]
	d ← (Reverse(s1)) >> cnt
		N, Z

LSL.[2/4] d, s1, s2
LSL.[2/4] d, s1, #cnt
	Logical Shift Left.
	d ← s1[31–s2:0] || 0^{s2};
	d ← s1[31–cnt:0] || 0^{cnt};
	If 's1' is a 16-bit operand it is sign-extended to 32 bits before the operation; 'd' must be a data register
		N, Z

LSR.[2/4] d, s1, s2
LSR.[2/4] d, s1, #cnt
	Logical Shift Right.
	d ← 0^{s2} || s1[31:s2];
	d ← 0^{cnt} || s1[31:cnt];
	If 's1' is a 16-bit operand it is sign-extended to 32 bits before the operation; 'd' must be a data register.
		N, Z

MERGE d, s1, s2
MERGE d, s1, #imm
	Bitwise Merge. This is a 32-bit merge that operates as follows:
	d ← (s1 & msk) | (s2 & !msk);
	d ← (s1 & msk) | (imm & !msk);
	'msk' is a 32-bit selection control mask, read from the SOURCE3 register. 's1' is a 32-bit operand. The second source operand may be either a data register or a 5-bit immediate value, zero-extended to 32 bits. 'd' must be a data register
		N, Z

SHFTD d, s1, s2
SHFTD d, s1, #imm
	Shift Double. 64-bit funnel shift that operates as follows:
	d ← ((SOURCE3 || s1) >> s2 Register[4:0]) [31:0]
	d ← ((SOURCE3 || s1) >> imm[4:0]) [31:0]
	SOURCE3 is a 32 bit operand. 's1' is a 32-bit operand; 'd' must be a data register.
		N, Z

Table 4-4 Shift and Bit-Field Instructions

Mnemonic & Operands	Description	Flags

SHMRG.1 d, s1, s2
SHMRG.1 d, s1, #imm
	Shift and Merge 1 byte.
	d ← (s2 << 8) | (s1 & 0x000000FF)
	d ← (imm << 8) | (s1 & 0x000000FF)
	's1' is an 8-bit operand; the second operand is a data register or 5-bit immediate. 'd' must be a data register
		N, Z

SHMRG.2 d, s1, s2
SHMRG.2 d, s1, #imm
	Shift and Merge 2 bytes.
	d ← (s2 << 16) | (s1 & 0x0000FFFF)
	d ← (imm << 16) | (s1 & 0x0000FFFF)
	's1' is a 16-bit operand; the second operand is a data register or 5-bit immediate. 'd' must be a data register
		N, Z

Notes: · "||" designates concatenation of bits.· "X^n": designates 'n' repetitions of bit 'X'.· "|" Indicates logical OR operation; "&" indicates logical AND operation; "!" indicates logical negation.· ">> N" indicates right shift by N bits.		

4.3.3 Single Bit Operations

Table 4-5 Single Bit Operations

Mnemonic & Operands	Description	Flags
BTST s1, s2 BTST s1, #bit_number	*Bit Test*. Test the bit indicated by the # bit_number field, of the 32-bit s1 operand and set the Z bits accordingly. The bit may be specified by the contents of the s2 D-register, or by a 5-bit immediate value. Z ← !s1[bit]	Z
BSET d, s1, #bit_number	*Bit Set*. Step 1: Test the bit in s1 indicated by the #bit_number field and set Z bits accordingly (Z ← !bit). Step 2: Set the selected bit in d.	Z
BCLR d, s1, #bit_number	*Bit Clear*. Step 1: Test the bit in s1 indicated by the #bit_number field and set Z bits accordingly (Z ← !bit). Step 2: Clear the selected bit in d.	Z
Notes: 1. BSET and BCLR: source and destination operands are typically the same, but this is not required.		

4.3.4 Data Movement And Extension Instructions

Table 4-6 Data Movement And Extension Instructions

Mnemonic & Operands		Description	Flags

MOVEI	d,#imm16	*Move 16-Bit Immediate Value*:
		if (destination is a directly addressed register)
		then
		    Destination ← SE32(imm16);
		else
		    Destination ← imm16; (2 byte destination size)
			-

MOVEAI	a,#imm24	*Move 24-bit Immediate Value* into bits [30:7] of the target address register. Bits 31 and [6:0] are cleared	-
EXT.1	d,s1	*Sign-Extend byte* from source to 32-bits and store result to destination. d ← s1[7]24 || s1[7:0] The sign extension is effective only if the destination is a register. If it is a memory destination, the destination size, being the same as the source size, makes the operation equivalent to MOVE.1. If the source is a direct register, 8 bits is extracted from that register and sign-extended.	N,Z
EXT.2	d,s1	*Sign-Extend 16 bit* source-1 operand to 32-bits and store result to destination. d ← s1[15]16 || s1[15:0] The sign extension is effective only if the destination is a register. If it is a memory destination, the destination size, being the same as the source size, makes the operation equivalent to MOVE.2. If the source is a direct register, 16 bits is extracted from that register and sign-extended.	N,Z
SETCSR	s1	*32-Bit Move*: CSR of current context ← s1 This is used to switch context back, since changing the destination context field of CSR is otherwise not possible to undo.	-
MOVE.1	d,s1	*8-Bit Move*: d ← s1 If the destination is a directly addressed register, then the upper 24 bits (31:8) are cleared to zero.	-
MOVE.2	d,s1	*16-Bit Move*: d ← s1 If the destination is a directly addressed register, then the upper 16 bits (31:16) are cleared to zero. Both source and destination addresses must be two-byte aligned – i.e., the least significant bit of address must be zero. Otherwise, result is architecturally undefined.	-
MOVE.4	d,s1	*32-Bit Move*: d ← s1 Moves a long word from source to destination. Both source and destination address must be quad-byte aligned – i.e., the two least significant bits of address must be zero. Otherwise, result is architecturally undefined.	-
Notes:· SE32: Indicates sign extension to 32 bits.			

4.3.5 Program Control Instructions

Table 4-7 Program Control Instructions

Mnemonic & Operands	Description	Flags
JMP<cc>.C.T/F offset	*PC-Relative Conditional Jump*: If specified condition function evaluates to '1' (True), then: PC ← PC + signed 23-bit offset; else: PC ← PC + 4. Branch prediction: T: Take branch (1); F: Continue (0). C: Condition Code Set Select: C = S : Select 16-bit Condition Codes (0); C = W : Select 32-bit Condition Codes (1). Offset is two's complement 23 bits in assembly syntax, of which only the most significant 21 bits are used in the instruction. The processor shifts this 21-bit offset value left by two bits before using it.	None
CALL An, offset	*PC-Relative Call To a Subroutine*: Step 1: An ← PC+4 Step 2: PC ← PC + Signed Offset Offset is two's complement 26 bits in assembly syntax, of which only the most significant 24 bits are used for the opcode. The processor shifts this 24-bit offset value left by two bits before using it. The return address is saved into the An register selected.	None
CALLI An, offset(Am)	*Address Indirect Call To a Subroutine*: Step 1: Target ← Am + Signed 18-bit Offset Step 2: An ← PC+4 Step 3: PC ← Target Offset is two's complement 18 bits in assembly syntax, of which only the most significant 16 bits are used for the opcode. The processor shifts this 16-bit offset value left by two bits before using it. The return address is saved into the An register selected.	None
RET s1	*Return from subroutine*: s1 → PC	None
SUSPEND	Suspends the current thread until an interrupt condition for that thread occurs. PC ← Address of next instruction MT_ACTIVE[thread] ← 0	None

BKPT s1	*Breakpoint Instruction*. The source operand is a bit mask that indicates which additional contexts to suspend. The bit number of a bit set in the mask is the thread number that is to be suspended. The thread that executes the BKPT instruction is always suspended.
	PC ← Address of BKPT instruction
	MT_DBG_ACTIVE[thread] ← 0
	For each thread specified by s1,
	MT_DBG_ACTIVE[specified thread] ← 0
	MT_BREAK[thread] ← 1
		None

4.3.6 Program Memory Access Instructions

Table 4-8 Program Memory Access Instructions

Mnemonic & Operands	Description	Flags
IWRITE pea_d, s1	(pea_d) ← s1 Writes the 32-bit source operand to pea address.	None
IREAD pea_s1	IREAD Register ← (pea_s1) Reads a shadow SRAM or flash program memory location into the issuing thread's IREAD_DATA register. The operand is always 32-bits wide.	None
Restriction: The destination (pea_d) and source (pea_s1) address modes are restricted to be one of the based (indirect) address modes – i.e., they are defined as:pea_d, pea_s1 = (An), or offset(An), or delta(An)++, (An)delta++, or (An, Dm)		
Notes: None of these program space instructions is blocking, i.e., they all take single-clock cycle to execute. The MEM BUSY bit in the context specific ROSR register indicates the completion of the operation. It is the program's responsibility not to initiate another of these instructions before the previously issued one completes.		

4.4 Instruction Formats and Encoding

Figure 4-2 shows the formats of instructions. The format codes of the first column provide cross-references for the instruction encodings in Table 4-9

Format		26			16	15	11	10		0	Class
	31 27	26	25 21	20	19 16	15	14 11	10 8	7 5	4 0	
1a	Opcode 5 Bits	Unused = 0 11 Bits				Opcode Extension 5 Bits		Unused = 0 11 Bits			No Operand
1b	Opcode 5 Bits	Unused = 0 11 Bits				Opcode Extension 5 Bits		Source-1 11 Bits			1-Operand Source
1c	Opcode 5 Bits	Destination 11 Bits				Opcode Extension 5 Bits		Unused = 0 11 Bits			1-Operand Destination
1d	Opcode 5 Bits	Destination 11 Bits				Opcode Extension 5 Bits		Source-1 11 Bits			2-Operand
2	Opcode 5 Bits	Destination 11 Bits				Bit Number 5Bits		Source-1 11 Bits			BSET, BCLR
3	Opcode 5 Bits	Destination 11 Bits				0	Source-2 Reg 4 Bits	Source-1 11 Bits			3-Operand General
4a	Opcode 5 Bits	0	Opcode Extension 5 Bits	0	Dn 4 Bits	Bit Num, Count 5 Bits		Source-1 11 Bits			3-Operand Restricted Dd=G1.op.imm
4b	Opcode 5 Bits	1	Opcode Extension 5 Bits	0	Dn 4 Bits	0	Source-2 Reg 4 Bits	Source-1 11 Bits			3-Operand Restricted Dd=G1.op.D2
5	Opcode 5 Bits	Immediate 16 Bits						Source-1 11 Bits			CMPI
6	Opcode 5 Bits	Destination 11 Bits				Immediate 16 Bits					Move Immediate
7	Opcode 5 Bits	Condi-tion 4 Bits	P C	Signed PC-Relative Offset O[20:0] 21 Bits							Conditional Branch P: Prediction Bit C: 16/32 CC select
8	Opcode 5 Bits	O[23:21] 3 Bits	An 3 Bits	Signed PC-Relative Offset O[20:0] 21 Bits							CALL, MOVEAI O: Offset Field
9	Opcode 5 Bits	O[15:13] 3 Bits	An 3 Bits	Offset O[12:8] 5 Bits		Opcode Extension 5 Bits		O[7:5] 3 Bits	Am 3 Bits	O[4:0] 5 Bits	CALLI O: Offset Field

Figure 4-2 Instruction Formats

Figure 4-3 shows the 11-bit encodings used in the 11 bit source1 and source2 instruction fields for the various addressing modes.

1	I_6	I_5	$\mathbf{A_2}$	$\mathbf{A_1}$	$\mathbf{A_0}$	I_4	I_3	I_2	I_1	I_0
0	**1**	**1**	$\mathbf{A_2}$	$\mathbf{A_1}$	$\mathbf{A_0}$	0	R_3	R_2	R_1	R_0
0	**1**	**0**	$\mathbf{A_2}$	$\mathbf{A_1}$	$\mathbf{A_0}$	m	I_3	I_2	I_1	I_0
0	**0**	**1**	8-BIT DIRECT ADDRESS							
0	**0**	**0**	8-BIT SIGNED IMMEDIATE							

I[6:0] – 7-bit unsigned immediate value that is left shifted by 0, 1, or 2 depending on operand size.
I[3:0] – 4-bit signed immediate value that is left shifted by 0, 1, or 2 depending on operand size.
M (Mode) – Selects post- (0), or pre- (1) addition of increment mode.
A[2:0] – 3-bit Address Register Selection.
R[3:0] – 4-bit Data Register Selection.
Direct Address – 8-bit Direct Address specifier, left shifted two bits to generate the byte address of a 32-bit register in the direct address space.
Signed Immediate – 8-bit immediate value that is sign-extended to 32 bits for use as an immediate operand.

Figure 4-3 Coding of Addressing Modes.

Table 4-9 specifies instruction formats and opcode assignments for all the instructions described in the preceding sections. Entries in the table are grouped by format, not by functionality. Refer back to Figure 4-2 for instruction formats. Opcode and opcode extension values are given in hexadecimal.

Undefined instruction encodings include unassigned opcodes and subops, and instruction bits set to 1 that are defined to be zero. The result of executing an undefined instruction is undefined.

Table 4-9 CPU Instruction Encodings

Type	Mnemonic	Opcodes$_{16}$		Notes
		Primary	Extension	
Format 1: 2-operand				
No-Operand Program Control	<reserved>	00	00	
	SUSPEND		01	
One-Operand Source	RET		04	
	IREAD		06	
	BKPT		07	
One-Operand Destination	<reserved>		05	1
Two-Operand Data Movement and Unary Operations	NOT.4		0A	
	NOT.2		0B	
	MOVE.4		0C	
	MOVE.2		0D	
	MOVE.1		0F	
	IWRITE		10	
	SETCSR		12	5
	EXT.2		15	
	EXT.1		17	
	LEA.4		1C	
	LEA.2		1D	
	LEA.1		1F	
	PDEC		1E	
	<reserved>	01	-	
Format 2: BSET, BCLR				
	BSET	04	-	
	BCLR	05	-	
	<reserved>	06, 07	-	
Format 3: 3-Operand, General				
	AND.2	08	-	
	AND.4	09	-	
	OR.2	0A	-	
	OR.4	0B	-	
	XOR.2	0C	-	
	XOR.4	0D	-	
	ADD.2	0E	-	
	ADD.4	0F	-	

Table 4-9 CPU Instruction Encodings

Type	Mnemonic	Opcodes$_{16}$		Notes
		Primary	Extension	
	ADDC	10	-	
	SUB.2	11	-	
	SUB.4	12	-	
	SUBC	13	-	
	<reserved>	14-17	-	
Format 4: 3-Operand, Restricted				
	MULS	02	01	2
	MULU		03	2
	MULF		05	2
	BTST		06	3
	CRCGEN		08	2
	MAC		09	2
	LSL.4		10	
	LSL.2		11	
	LSR.4		12	
	LSR.2		13	
	ASR.4		14	
	ASR.2		15	
	BFEXTU		16	
	BFRVRS		18	
	SHFTD		1A	
	MERGE		1C	
	SHMRG.2		1E	
	SHMRG.1		1F	
	<reserved>	03	-	
Format 5: Compare Immediate				
	CMPI	18	-	3
Format 6: Move Immediate				
	MOVEI	19	-	4
Format 7: Conditional Branch				
	JMP<cc>	1A	-	
Format 8: CALL, MOVEAI				
	CALL	1B	-	
	MOVEAI	1C	-	

Table 4-9 CPU Instruction Encodings

Type	Mnemonic	Opcodes$_{16}$		Notes
		Primary	Extension	
	<reserved>	1D	-	
Format 9: Call Indirect				
	CALLI	1E	-	
	<reserved>	1F	-	

1. Dest operand must be a memory address mode (not immediate or register).
2. Destination is implicit MAC_HI, MAC_LO, MAC_RC16 registers.
   Destination field in instruction is coded as zero.
3. Destination is implicit CSR register (CCs).
4. If destination is data memory address, operand size is 16 bits.
5. Destination is coded with the direct register address of the CSR register.

5.0 Peripherals

5.1 Overview

The IP3023 supports a number of I/O ports, each of which can be configured as general purpose I/O or assigned to a specific function. The available hardware functions include support and control functions, such as clock generation and interrupts, as well as functions for allowing connections to specific on-chip I/O controller/accelerator units. Table 5-1 summarizes the functions available at each port.

Table 5-1 I/O Port Functions

Port	Function
A & B	Provide support for external instruction memory as well as some slave peripheral devices. To provide good performance, processor instruction fetches to flash and SDRAM accesses are controlled by hardware state machines. Accesses to all other devices attached to these ports are performed through direct manipulation of the I/O pins.
C	Provides hardware support for MII.
D	Provides a second MII port.
E	Provides ½ of a third MII port as well as SerDes hardware functionality. This SerDes block does not support 10Base-T, because the chip itself implements only one receiver, which is associated with Port F. Also provides a high-speed GPSI port.
F	Provides the other half of the MII (Port E also supporting half) and SerDes hardware functionality. This SerDes supports 10Base-T.
G	Entirely GPIO with no additional hardware I/O acceleration.
H	Provides a fourth MII port as well as GPIO and hardware clock generation support for peripheral timing.

5.2 Shared Port Architecture

All ports share a common base architecture. Depending on the specific requirements of the port, each port can be composed of :

- A register set to control and monitor the port functions.
- Data FIFOs for efficient movement of transmit and receive data.
- One or more independently operating function blocks.
- Muxes that control function selection.

All ports share these common features:

- Configurable (except port A) to operate as GPIO only.
- FIFO interface for data, independent of protocol. Alignment and packing of data within the 32-bit FIFO word is function specific. FIFOs are up to 128 bytes large; the actual size is port specific.
- External clock sources available.

5.2.1 Port Registers

Port registers are used for function selection, setting function parameters, function control, and reporting function and port status. Port specific register definitions are provided in Section 6.6 and following sections.

5.2.2 Interrupts

Each port has four registers that participate in interrupt control for that port:

- Interrupt Status Register
- Interrupt Mask Register
- Interrupt Set Register
- Interrupt Clear Register

These are 32-bit registers, but only bits [15:0] are dedicated to interrupt control. Bits 15:0 correspond by bit position to each of 16 potential interrupt sources.

For each port, there are three bits in the global interrupt register INT_STAT1[23:0] and in each of the per-thread interrupt mask registers INT_MASK1[23:0]. The three interrupt bits correspond to port interrupt conditions as follows:

- Interrupt 0 – Receive FIFO high watermark condition.
- Interrupt 1 – Transmit FIFO low watermark condition.
- Interrupt 2 – All other port interrupt conditions.

5.2.3 FIFO Management

The FIFOs are initialized by asserting one of the set bits TX FIFO Reset or RX FIFO Reset.

The TX Underflow and RX Overflow Interrupts are provided to inform software of attempts to transmit when the transmit FIFO is empty or to receive when the receive FIFO is full.

At any time, software can determine how full the transmit and receive FIFOs are by examining the TX_FIFO_DEPTH and RX_FIFO_DEPTH registers.

To help avoid overflow or underflow conditions, the concept of FIFO watermarks is implemented. Software uses control registers to set the TX FIFO Low Watermark level and RX FIFO High Watermark level. Hardware uses the RX FIFO High Watermark interrupt or TX FIFO Low Watermark interrupt to inform software when a watermark level has been reached.

If a transmit FIFO Watermark Interrupt, or a receive FIFO Watermark Interrupt has occurred, and the appropriate action has not been taken to address the interrupt (i.e., fill the transmit FIFO or empty the receive FIFO) the interrupt will persist against any attempt by software to clear the interrupt. (The mask bit must be set to cause an interrupt.)

5.2.3.1 Receive FIFO Selection, Behavior and Restrictions

A second receive FIFO is installed in those I/O ports that have a function needing a second FIFO (see Table 5-2). Where only one receive FIFO is installed in an I/O port, the receive FIFO is installed as FIFO 0.

Access to a specific receive FIFO is controlled through the Receive FIFO Select bit (Function Select[3]). Setting this bit to 0 causes all accesses to the receive FIFO (reset, status, data) to reference receive FIFO 0. Setting this bit to 1 causes receive FIFO accesses to reference receive FIFO 1.

An interrupt that was set due to the action of one FIFO will remain set after the Receive FIFO Select bit is changed to select the other FIFO, even if the newly selected FIFO is not asserting this interrupt. If a FIFO is filled beyond the level set by the watermark trigger level and that FIFO is not currently selected, an interrupt will not be asserted until that FIFO has been selected through the Receive FIFO Select bit. The following register fields are affected by Receive FIFO Select bit:

Interrupt status[13]	RX FIFO Overflow interrupt
Interrupt status[12]	RX FIFO Watermark interrupt
Interrupt status[21:16]	RX FIFO level
Interrupt set[30]	RX FIFO reset
RX FIFO[31:0]	RX FIFO data

When a receive FIFO overflows or a transmit FIFO underflows, all other FIFO status information is undefined, and software must reset the FIFO.

Table 5-2 Port Function Summary

Port #	Port Width	External Int	Clock Output	TX FIFO Size	RX FIFO Size	Function 0	Function 1	Function 2	Function 3
A	32	no	no	64 Byte	64 Byte	Flash	SDRAM	N/A	GPIO
B	8	yes	yes	128 Byte	128 Byte	Flash	SDRAM, Flash, Clock	GPIO	N/A
C	16	yes	no	64 Byte	64 Byte	GPIO	MII *	N/A	N/A
D	18	yes	no	64 Byte	64 Byte	GPIO	MII *	N/A	N/A
E	8	yes	no	64 Byte	64 Byte	GPIO	SerDes	MII *	N/A
F	8	yes	no	64 Byte	64 Byte	GPIO	SerDes	MII *	N/A
G	32	yes	no	N/A	N/A	GPIO	N/A	N/A	N/A
H	16	yes	yes	64 Byte	64 Byte	GPIO	MII *	GPIO, Clock	N/A

* A second FIFO is available at these ports.

5.3 External Flash Controller

This controller manages the interface between the IP3023 and an the external flash device. The controller can respond to the IREAD instruction of the processor. The controller can also respond to a more general register interface on Port A and Port B.

This controller is compatible with any 8-bit flash device that has a JEDEC standard non-multiplexed address path of 22-bits and which is controlled by the $\overline{CE}$, $\overline{OE}$, and $\overline{WE}$ pins. This includes the following devices:

Intel 28FxxxJ3 compatible devices
Intel 28FxxxS3 compatible devices
Intel 28FxxxB3 compatible devices
AMD Am29LV compatible devices
AMD Am29DL compatible devices
AMD Am29PL compatible devices

Table 5-3 shows the signals and pins that interface with the external flash device.

The flash device is physically controlled by sending external bus commands which are combinations of the $\overline{FCE}$, $\overline{OE}$, and $\overline{WE}$ signals; those supported by this controller are shown in Table 5-4. This list is common to all flash types supported by this controller. These external bus commands are generated automatically by the flash controller in response to IREAD instructions or to user commands that have been sent to it.

The default state is for the controller to hold the flash device in standby mode. This allows the same pins on IP3023 to be shared safely with the SDRAM controller or other functions on the same port.

Table 5-3 External Flash Signal Summary

Flash Signal	Pin	Direc-tion	Description
ADDR [21:0]	PA[21:0]	O	Byte address on the external flash device.
$\overline{OE}$	PA22	O	Output enable.
$\overline{WE}$	PA23	O	Write enable.
DATA [7:0]	PA[31:24]	I/O	Input and output data.
$\overline{FCE}$	PB7	O	Flash Chip Enable. Connected to flash device's $\overline{CE}$ pin. It is active only when the flash function for Port A is enabled. This pin should be tied to a pullup resistor, so that the flash chip isn't enabled while the IP3023 is in reset.

Table 5-4 Supported Flash Bus Commands

Command	$\overline{FCE}$	$\overline{OE}$	$\overline{WE}$	A[21:0]	DQ[7:0]
Standby	H	--	--	--	--
Output Disabled	L	H	H	--	--
Read	L	L	H	Address	Data In
Write	L	H	L	Command or Address	Command or Data Out
Illegal	L	L	L	--	--

5.3.1 Processor IREAD Interface

The controller responds to the IREAD instruction of the processor, returning a word from the desired address.

The IREAD instruction is handled only when this port's FUNC_SEL is set to the flash function (the default power-up condition). If another function on the same port is selected later, the flash controller is disabled automatically ($\overline{FCE}$ is deactivated). This means that the IP3023 can boot by performing the initial fetches from flash without having to explicitly enable this port function. It also means that, to copy data from the flash to the SDRAM (which share the same physical pins), software must alternate between enabling these two functions on the port.

When disabled (Port B flash function not selected), the controller completely ignores all IREAD instructions, and does not send any response; instead it asserts an error interrupt and enters an error state, where it remains until function reset. The error condition should be cleared before applying the function reset; otherwise, the controller simply re-enters the error state.

IREAD instructions cause the flash controller to automatically send four successive read bus commands each returning a byte to an internal buffer. These bytes are combined together to form a full 32-bit word that is returned to the processor core.

The read/write access timing parameter (XFL_TAVAV) defaults to 64 CORE_CLK cycles to guarantee that any device rated faster than 150ns can be supported on future IP3000 products rated up to a maximum frequency of 400MHz. This parameter is configurable to lower values to speed up the flash interface.

The number of cycles required to complete an instruction fetch can be calculated using the formula: cycles = 4*(XFL_TWAIT + XFL_TAVAV) + 6 + PIPE_DELAY. Unfortunately, the value of PIPE_DELAY cannot be specified, since it depends on the scheduling table configured by software. For 100% scheduling of the requesting thread, the value would be six.

If software disables the controller (by deasserting FUNC_SEL) while an IREAD operation is in progress, then the flash controller enters ERROR state. Because the operation has not completed, there is never any acknowledgement sent back to the processor core. Without this signal, the MEM_BUSY bit in the thread-specific ROSR register cannot be cleared. The only way to exit the ERROR state is to reset the controller. By ensuring that FUNC_SEL is reasserted before applying reset to the controller, the original IREAD operation will be restarted.

If an IREAD is interrupted by software asserting XFL_EN in preparation for an XFL-style operation, then the flash controller can perform internal arbitration. It allows the IREAD to complete successfully before switching over to handle XFL-style user commands. Any XFL_START pulse sent during this period when the IREAD is completing will be ignored. Once the flash controller enters the XFL-mode, it ignores subsequent IREAD requests until XFL_EN is deasserted.

5.3.2 Port Register Interface

The IWRITE instruction is not directly supported by this controller. Instead, a port interface is provided that is connected just like any other I/O function, with user commands, data FIFOs, and register parameters. Table 5-5 lists the user commands that are available through direct port access.

Table 5-6 shows the register fields through which software programs the flash controller.

Table 5-5 Supported User Commands

CMD Code	User Command	Function
11	XFL_NOP	Does nothing. Sends "output disabled" bus command.
01	XFL_READ	Reads a byte of data from the flash at the specified byte address. Used to read both data and status of the flash device. Sends “read” bus command.
10	XFL_WRITE	Writes a byte of data to the flash at the specified byte-address. Used to transfer commands and data directly to the device. Sends “write” bus command.
00	XFL_ILLEGAL	The flash controller simply passes the user command directly to the OE# and WE# pins. This combination is possible but illegal for all compatible flash devices.

Table 5-6 External Flash Register Interface

Field Name	Register	Read/ Write	Description
XFL_ERR	Interrupt	RO	This signal is held high by the flash controller to indicate that it is in the error state and that it must be reset before it can continue with any other operation.
XFL_DONE	Interrupt	RO	This interrupt is asserted by the flash controller to indicate that it has completed the user command successfully. It is not asserted when an error condition occurs. It is only asserted for user commands, and not for IREAD (the MEM_BUSY bit in the ROSR can be used for those instructions).
XFL_START	Set	WO	Writing a 1 to this bit causes the controller to execute the command stored in XFL_RW_CADQ.
XFL_EN	Control 0	R/W	Enable direct low-level flash access through the port. Must be enabled in order to respond to XFL_START pulses. This automatically disables the IREAD mechanism when enabled. If this signal becomes active during an IREAD that is already in progress, the older fetch is allowed to finish before the port is disabled.
XFL_TWAIT[1:0]	Control 0	R/W	Number of wait states to insert between $\overline{\text{FCE}}$ low and $\overline{\text{WE}}$/$\overline{\text{OE}}$ low. Changes to XFL_TWAIT always take effect on the next XFL_START pulse or for the next IREAD instruction. **XFL_TWAIT Wait Cycles** 00 4 01 1 10 2 11 3
XFL_TAVAV[5:0]	Control 0	R/W	Read/write access time expressed in core clock cycles. Changes to XFL_TAVAV[5:0] during the first XFL_TWAIT[1:0] cycles of the external flash device access take effect on this access. Changes outside this brief period take effect on the next XFL_START pulse or for the next IREAD instruction. **XFL_TAVAV Cycles** 0 64 1 0 2... XFL_TAVAV–1
XFL_ACT	Status 0	RO	The flash controller sets this bit to indicate that it is active. Software should ensure that the controller is not active before sending a command.
XFL_RD_DATA[7:0]	RX FIFO	RO	Byte of data that has been read from the flash device.
XFL_RW_CADQ[31:0]	TX FIFO	RO	Read/Write Command+Address+Data: 31:30 CMD1:0 Flash user command 29:8 ADDR21:0 22 bits of address 7:0 DATA7:0 8 bits of data

5.3.3 Configuring the Flash Controller

After reset, the external flash memory controller is set to operate with the most conservative values for access and wait state timing. The flash controller access time can then be modified to optimize the performance for the current operating speed. It is the responsibility of the software to ensure that flash timing parameters are not violated, especially when preparing to change the system clock.

5.3.4 Using the Flash Controller

The XFL_EN control bit must be enabled for the controller to respond to user commands. Programs initiate user commands by loading a command and related parameters into XFL_RW_CADQ**,** then writing a 1 to XFL_START**.**

Any XFL_START pulse sent while the flash controller is busy on an earlier operation is automatically ignored. Software should check the status of XFL_ACT.

If software disables the controller (by deasserting FUNC_SEL) while an XFL-style user command is in progress, the flash controller enters ERROR state. The only way to exit the ERROR state is to reset the controller.

If software disables the controller while it is idle, it enters a deselected state. Any IREAD request or XFL_START pulse that occurs while in this state causes the controller to enter the ERROR state.

Flash devices do not signal the completion of program/erase commands back to the controller; so software must poll the flash device periodically (for example, every 10 µs) using regular XFL_READ user commands. The method for detecting completion is device dependent.

5.3.4.1 Flash Read Command

XFL_READ user commands read only one byte from the external flash device at the speed specified in the read access timing parameter (XFL_TAVAV). This is a lower-level access to the device than the IREAD command, since no internal state machine is invoked to perform four of these operations in succession.

The number of cycles required to complete the XFL_READ operation can be calculated using the formula: XFL_TWAIT + XFL_TAVAV + 4 + PIPE_DELAY. Again, the value of PIPE_DELAY cannot be specified since it depends on the scheduling table configured by software.

The external flash device bus commands generated for these IREAD instructions or XFL_READ user commands are identical for both AMD and Intel flash devices -- they start the operation when both $\overline{\text{CE}}$ and $\overline{\text{OE}}$ go active low.

The complete sequence is as follows (refer to):

1. If required, software pre-programs a faster read access cycle time into the XFL_TAVAV register and a smaller value for the number of wait states.
2. Software must ensure that the flash controller function is selected for this port.
3. Software must enable the low-level interface by ensuring XFL_EN is asserted.
4. Software writes a word to the output FIFO.
5. Software initiates transfer of the above word from the FIFO to the flash controller by setting XFL_START.
6. The controller asserts XFL_ACT to indicate that it is busy.
7. The controller asserts $\overline{\text{FCE}}$ and drives address bits to the flash device.
8. After XFL_TWAIT cycles the controller asserts $\overline{\text{OE}}$ to start the READ bus command operation.
9. The controller waits for XFL_TAVAV core clock cycles.
10. The controller captures the data byte on the input bus.
11. On the next cycle, the controller pulses XFL_RD_VLD to indicate that the data is available on the output bus and XFL_WR_ACK to indicate that the command/address/data word can be removed from the FIFO. In parallel, it deasserts $\overline{\text{OE}}$ and $\overline{\text{FCE}}$ to disable the flash-device.
12. On the next cycle, the flash-controller goes to idle and clears XFL_ACT to indicate that it is no longer busy.
13. Software should then deassert XFL_EN to switch off this low-level interface.

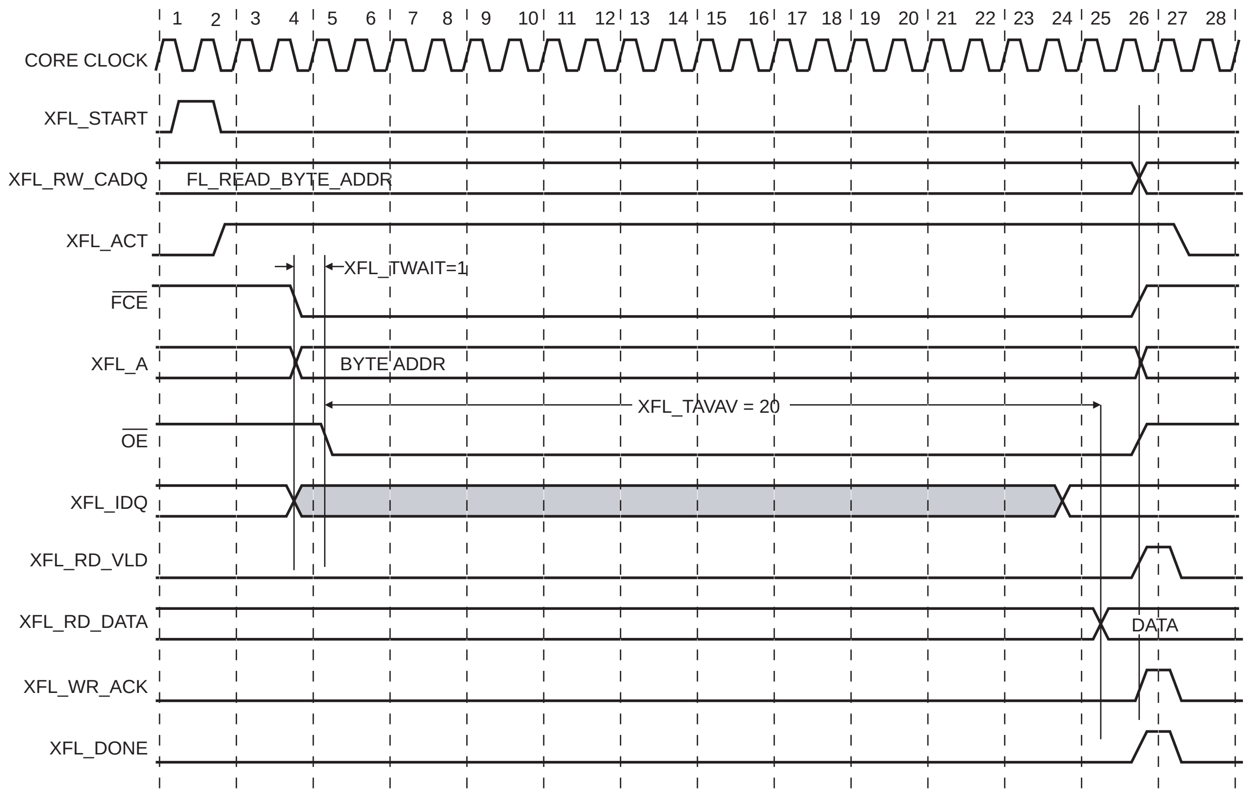


Figure 5-1 XFL_READ Sequence

5.3.4.2 Flash Write Command

XFL_WRITE user commands cause the flash controller to write bus commands to the flash device that may be either instructions or data. This command does not “program” the flash device directly. Instead, a sequence of commands is sent to the device, followed by the required data. The actual command sequence varies depending on the type of flash device that is connected. Intel and AMD devices have very different command sequences. Refer to the device’s data sheet for details.

The number of cycles required to complete the XFL_WRITE operation can be calculated using the formula: XFL_TWAIT + (2*XFL_TAVAV) + 3 + PIPE_DELAY. Yet again, the value of PIPE_DELAY cannot be specified, since it depends on the scheduling table configured by software.

1. If required, software pre-programs a faster read access cycle time into the XFL_TAVAV register and a smaller value for the number of wait states.
2. Software must ensure that the flash controller function is selected for this port.
3. Software must enable the low-level interface by setting XFL_EN .
4. Software writes a word to the output FIFO.
5. Software initiates transfer of the above word from FIFO to flash controller by setting XFL_START.
6. The controller asserts XFL_ACT to indicate that it is busy.
7. The controller enables write mode of tri-state drivers by asserting XFL_EN_$\overline{\text{WR.}}$
8. The controller asserts $\overline{\text{FCE}}$ and drives address bits to the flash device.
9. After XFL_TWAIT cycles the controller asserts $\overline{\text{WE}}$ to start the write operation.
10. The controller waits for XFL_TAVAV core clock cycles.
11. The controller deasserts $\overline{\text{WE}}$ to continue with the second half of the write operation.
12. The controller again waits for XFL_TAVAV core clock cycles.
13. The Controller signals XFL_WR_ACK to indicate that the command/address/data word can be removed from the FIFO, because it has been written to the flash device.
14. On the next cycle, the flash controller deasserts XFL_EN_WR to disable tri-state drivers and at the same time clears XFL_ACT to indicate that it is no longer busy.
15. Software should deassert XFL_EN to switch off this low-level interface.

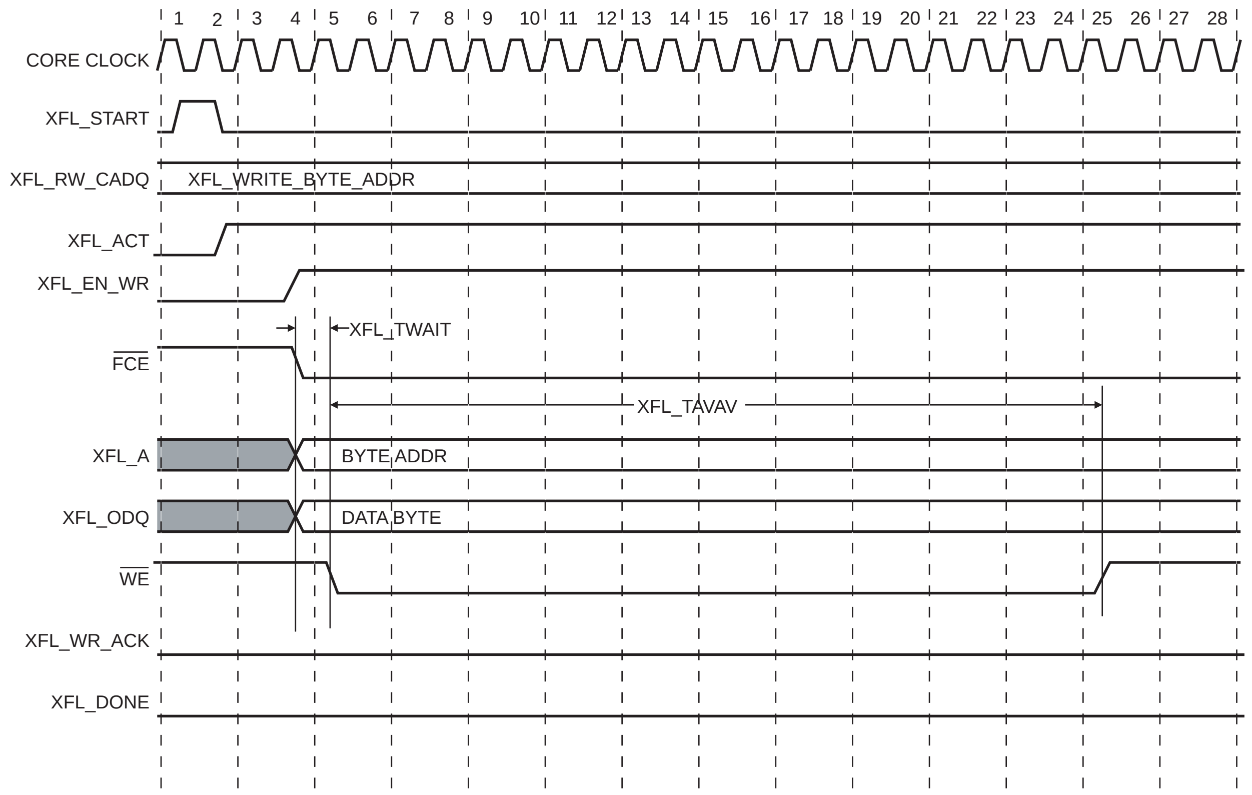


Figure 5-2 XFL_WRITE Sequence Start

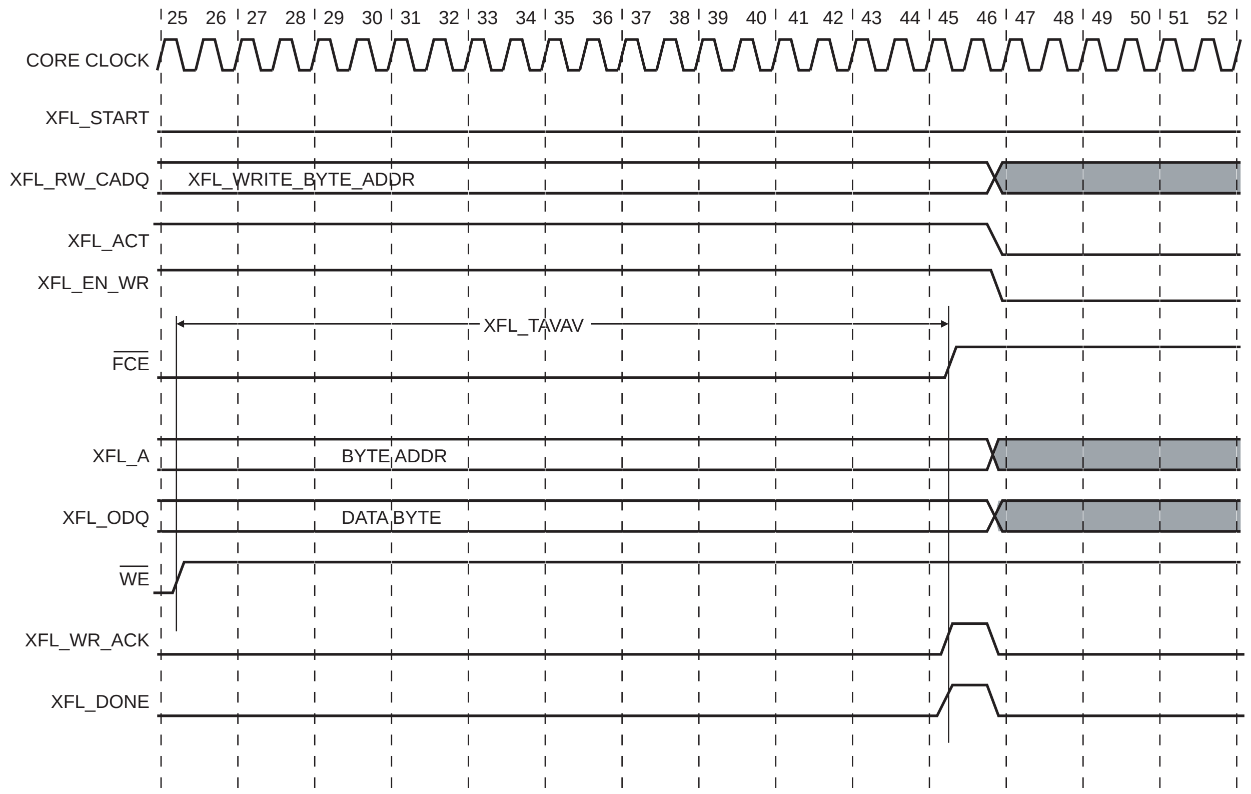


Figure 5-3 XFL_WRITE Sequence End

5.3.4.3 Flash NOP Command

XFL_NOP user commands do nothing interesting on the external flash device. The command is provided as a means for selecting $\overline{\text{FCE}}$ while holding $\overline{\text{WE}}$ & $\overline{\text{OE}}$ inactive.

5.3.4.4 Flash Illegal Command

The XFL_ILLEGAL command corresponds to the state when both the $\overline{\text{OE}}$ and $\overline{\text{WE}}$ signals are active simultaneously. This is an illegal state for every one of the compatible flash devices listed previously, but may be useful for some device control functions. Because the interface provides direct access to the $\overline{\text{OE}}$ and $\overline{\text{WE}}$ pins connected directly to the flash device, it is possible to set them in this state. The flash controller simply passes the desired value along to the external device without any decode.

5.3.5 Switching Between Flash and SDRAM

When both flash and SDRAM devices are in simultaneous use, Port B remains set to Function 1 (Flash + SDRAM), while software switches Port A between the flash and SDRAM.

To switch from flash to SDRAM, change the Port A function select from 0 (flash) to 1 (SDRAM).

To switch from SDRAM to flash:

1. Change Port A function select from 1 (SDRAM) to 0 (flash).
2. Send a function reset to the flash controller. This restarts any IREAD sequences which were interrupted when switching over to SDRAM. The reset is required; otherwise, subsequent IREAD requests would stall behind the interrupted one.

5.4 SDRAM Controller

The SDRAM controller manages the interface between the IP3023 and an external SDRAM data memory. The controller connects to the port data FIFOs just as any other I/O function; therefore, software can set the FIFO watermark at the desired level to receive an interrupt when the transfer has completed. High-level user commands are provided to control the SDRAM device.

Table 5-7 shows the signals used by the SDRAM controller. Some of the signals use pins of Port A, while others use Port B pins.

Table 5-8 shows the register fields through which software programs the SDRAM controller.

Table 5-7 SDRAM Controller Signal Summary

Signal	Port Pin	Direction	Description
ADDR [9:0]	PA[9:0]	O	Address pins for read/write
ADDR [10] (AP)	PA10	O	Auto-Precharge Address[10]. Indicates auto-precharge during read/write commands and one/all banks during precharge command. Functions as ADDR[10] during Active bus commands.
ADDR [12:11]	PA[12:11]	O	Address pins for read/write.
BA [1:0]	PA[14:13]	O	Bank Select Address. Selects bank to be activated during row address latch time. Selects bank for read/write during column address latch time.
DQM	PA15	O	Data Mask. This is a single bit output which is connected to both the upper and lower data masks on the external SDRAM device. It is used to block extra words from being written at the end of a long transfer.
DATA [15:0]	PA[31:16]	I/O	Read and write data.
$\overline{\text{WE}}$	PB0	O	Write Enable. Enables write operation and row precharge. Latches data-in starting from $\overline{\text{CAS}}$, $\overline{\text{WE}}$ active.
$\overline{\text{RAS}}$	PB1	O	Row Address Strobe. Latches row addresses on the positive going edge of the CLK with $\overline{\text{RAS}}$ low. Enables row access and precharge.
$\overline{\text{CAS}}$	PB2	O	Column Address Strobe. Latches column addresses on the positive going edge of the CLK with $\overline{\text{CAS}}$ low. Enables column access.
$\overline{\text{CS}}$	PB3	O	Chip Select. Disables or enables device operation by masking all inputs except SD_CLK, and CKE.
CKE	PB4	O	Clock Enable. Masks system clock to freeze operation on the next clock cycle. CKE should be enabled at least one cycle prior to new command.
SD_CLKI	PB5	I	Input for driving SDRAM clock. Can be provided by external clock or can be tied to SD_CLKO.
SD_CLKO	PB6	O	Reference clock for external SDRAM. It is generated by dividing the core clock. Thus, the nominal value for the 250MHz rated device is 62.5MHz (250MHz/4), which is enough to guarantee a throughput of 100MB/s. However, since the core clock frequency of the processor is variable, this value must change to match.

Table 5-8 SDRAM Controller Register Interface

Field Name	Register	Read/ Write	Description
SD_ERROR	Interrupt	RO	Indicates that the SDRAM controller is in an error state and must be reset. This condition occurs when the controller receives an SD_START, but the SDRAM function on Port B is not enabled.
SD_DONE	Interrupt	RO	Indicates that the SDRAM controller has completed the current transaction. This interrupt occurs only for user commands, not for auto-refresh sequences. Occurs on SD_ACT high to low transition.
SD_START	Set	WO	Writing a 1 here causes the SDRAM controller to execute the command in the SD_CMD register. Up to four commands can be queued while the controller is still busy operating on an earlier one, but hardware does no checks to ensure that the FIFO depth of four is not exceeded. Programming Note: This bit must be set after all the parameters are configured. The precise number of cycles to wait can be calculated by rounding up the result of: (SD_CLK ns / CORE_CLK ns)+1. This allows the asynchronous boundary to be crossed safely.
SD_CL[1:0]	Control 0	R/W	CAS Latency expressed in number of SD_CLK cycles. This register must be programmed during initialization; otherwise, data transfer cannot occur. Behavior is undefined if this register is modified during a data transfer. **SD_CL** **CAS Latency** 01 1 10 2 11 3 00 4
SD_TRDL[1:0]	Control 0	R/W	Last data into PRE.A. Max (tRDL, (tDAL–tRP)) expressed in SD_CLK cycles, with any fractions rounded up. tRDL = last data into SDRAM to PRE.A command tDAL = last data into SDRAM to ACT command tRP = row precharge time of SDRAM. Must be programmed during initialization; otherwise, data transfer cannot occur. Behavior is undefined if modified during a data transfer.
SD_TRCD[2:0]	Control 0	R/W	RAS to CAS Delay expressed in number of SD_CLK cycles. To ensure correct operation, this parameter should be set to the value of: int(tRCD/sd_clk + 1) Fractions cause the value to be rounded up. Must be programmed during initialization; otherwise, data transfer cannot occur. Behavior is undefined if modified during a data transfer.
SD_TRP[2:0]	Control 0	R/W	Row Precharge Time expressed in number of SD_CLK cycles. To ensure correct operation, this parameter should be set to the value of: max(2,int(tRP/sd_clk + 1)) The minimum supported value is 2; fractions cause the value to be rounded up. Must be programmed during initialization; otherwise, data transfer cannot occur. Behavior is undefined if modified during a data transfer.

Table 5-8 SDRAM Controller Register Interface

Field Name	Register	Read/ Write	Description
SD_TRAS[3:0]	Control 0	R/W	Min time from ACT to PRE.A command expressed in SD_CLK cycles. To ensure correct operation, this parameter should be set to the value of: int(tRAS/sd_clk + 1) Fractions cause the value to be rounded up. Must be programmed during initialization; otherwise, data transfer cannot occur. Behavior is undefined if modified during a data transfer.
SD_TRFC[3:0]	Control 0	R/W	Refresh Time. Max (tRC, tRFC) expressed in SD_CLK cycles tRC = ACT to ACT command period tRFC = duration of refresh sequence To ensure correct operation, this parameter should be set to the value of: int(max(tRC,tRFC)/sd_clk + 1) i.e., the maximum value of tRC or tRFC expressed in SD_CLK cycles with any fractions rounded up. This register must be programmed during initialization; otherwise, data loss will occur. Behavior is undefined if modified during a data transfer.
SD_REF[15:3]	Control 0	R/W	Refresh Period. Frequency with which auto-refresh commands must be sent to SDRAM expressed in SD_CLK cycles. The refresh frequency is treated as a 16-bit value, of which the low-order 3 bits are zero. Refer to programming notes in Table 5-12.
SD_EN_REF	Control 0	R/W	Enable Auto Refresh.
SD_ADDR[26:2]	Control 1	R/W	Word address at which to perform read/write on external SDRAM. Must be word-aligned.
SD_WORD_CNT[6:0]	Control 2	R/W	Desired length for read and write accesses to the SDRAM expressed in number of words. This parameter is ignored for other user commands. Refer to Table 5-9 for values.
SD_CMD[2:0]	Control 2	R/W	SDRAM controller command. Refer to Table 5-10 for values.
SD_ACT	Status 0	RO	Indicates that the SDRAM controller is busy processing a user command. It is not asserted when a hardware generated auto-refresh is in progress.

Table 5-9 SDRAM Word-Count Values

SD_WORD_CNT	Words	Transfers (16-bit bus)
0000001	1	2
0000010	2	4
0000011	3	6
…	…	…
1111111	127	254
0000000	128	256

The SDRAM is physically controlled by sending external bus commands that are combinations of the CKE, $\overline{\text{CS}}$, $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WE}}$, and A10 signals. To simplify the interface presented to the programmer, the SDRAM controller generates these external bus commands in response to user commands that are sent to it, as Table 5-10 shows.

Table 5-10 SDRAM Commands

SD_ CMD	Command	Effect
000	*null*	No-action
001	SD_READ	Read sequence. Read a user-specified number of words from the external SDRAM.
010	SD_WRITE	Write sequence. Write a user-specified number of words to the external SDRAM.
011	SD_NOP	Nop sequence. Send the NOP bus command that does nothing, but is required for SDRAM device initialization and to awaken the device from self-refresh (“sleep”) mode.
100	*reserved*	Enters error state
101	SD_SLEEP	Sleep sequence. Force the external SDRAM to enter self-refresh mode that maintains the contents of the memory without any external clock.
110	SD_INIT	Init sequence. Send sequence of commands required on device reset and initialize the mode register on the external SDRAM with user-specified values.
111	<reserved>	Enters error state

5.4.1 Address Translation

This controller automatically performs address translation from linear word addresses supplied by the software to bank/row/col addresses required by the external SDRAM. The following table indicates the supported SDRAM structures and capacities.

Config	Banks	Rows	Cols	Capacity (16-bit)
1	1	4K	256	16 Mb
2	2	4K	256	32 Mb
3	4	4K	256	64 Mb
4	4	4K	512	128 Mb
5	4	8K	512	256 Mb
6	4	8K	1024	512 Mb

The controller needs no parameter to select one of the above configurations, due to the method by which linear word addresses are translated into physical bank/row/col for the SDRAM, as Figure 5-4 shows.

The address range SD_ADDR[20:2] is common for all SDRAM configurations and all software can safely use this address space without any concerns. The addresses above this range can be utilized only if the SDRAM device has a large enough capacity; the hardware makes no check on the addresses presented nor to the characteristics of the device connected to it. The system software must be aware of these details.

The controller automatically increments the column address (col[9:0]) in multiword transfers which could result in bits within col[9:8] being set even when the SDRAM device does not have that many columns. The resulting behavior is specific to the actual device.

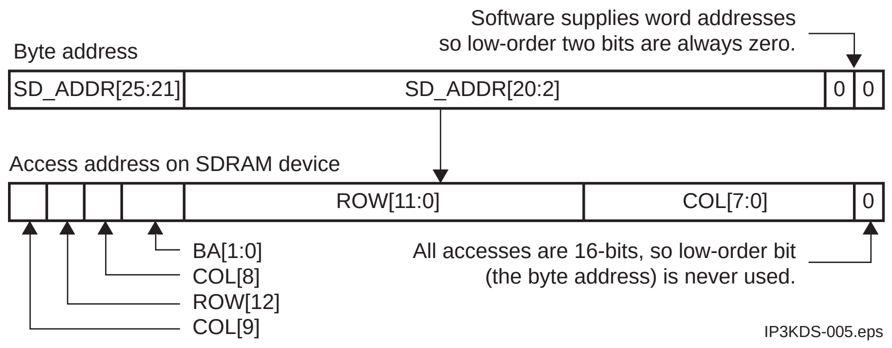

Software supplies word addresses so low-order two bits are always zero.
Byte address
SD_ADDR[25:21]
SD_ADDR[20:2]
0
0
Access address on SDRAM device
ROW[11:0]
COL[7:0]
0
BA[1:0]
COL[8]
ROW[12]
COL[9]
All accesses are 16-bits, so low-order bit (the byte address) is never used.
IP3KDS-005.eps

Figure 5-4 SDRAM Address Translation

5.4.2 Configuring the SDRAM Controller

SDRAM requires a complicated power-on sequence to guarantee correct operation of the device. The JEDEC standard recommends the following:

1. Apply power and start clocks.
2. Clock must stabilize within 100ms after power stabilizes.
3. Hold the following control pins inactive; SD_CS=1 and SD_CKE=0, for a minimum of 1ms after supply voltage reaches the desired level. This combination of control signals is the reset state for the SDRAM controller.
4. Software should issue one SD_NOP command while auto-refresh is disabled. This effectively activates the SD_CKE pin.
5. Software pauses 200ms after this SD_NOP.
6. Software configures the SD_REFRESH register with the required value and enables the auto-refresh feature of the SDRAM controller.
7. Issue precharge bus commands for all banks of the device.
8. Issue eight or more auto-refresh bus commands.
9. The internal SDRAM mode register can now be configured using the Mode Register Set bus command. This register specifies the CAS latency, burst-length and burst-type for the part.

The command SD_INIT is provided as a convenient way to apply the sequence of bus commands required in steps #7, #8 and #9 above.

SDRAM needs to be periodically refreshed. This controller takes care of sending auto-refresh commands to the SDRAM as required so that no intervention is required from the programmer, other than to set up the refresh period and command duration in the configuration registers. For the auto-refresh commands to take effect, the SD_CKE pin must be high (device not in sleep mode). This is normally achieved by sending an SD_INIT command to initialize the device or by sending an SD_NOP command to exit sleep mode.

The SDRAM specification states that banks must be opened before any read or write activity by using the active (ACT) bus command with the desired bank and row address being presented simultaneously. To simplify the protocol for the programmer the SDRAM controller automatically applies these ACT commands as required on read and write accesses; therefore, no user command has been provided for this function. Additionally the protocol automatically closes the bank (using one of the precharge commands) after the transfer has completed; there is no optimization to try to keep banks open for sequential burst accesses. So, given a single read/write user command, the actual sequence of signals interfacing to the external SDRAM device would look something like Figure 5-5.

Note that the earliest time the auto precharge-all (PRE.A) can be issued after the final READ is transfer-size+(CL-2) as indicated by the “IDLE” cycle in Figure 5-5.

Two other parameters must be matched to ensure correct SDRAM behavior:

- The minimum time from one ACT to the next is specified in the data sheets as tRC. This controller combines the greater of tRFC and tRC into a single user-specified parameter "tRFC".
- The minimum time from an ACT to a PRE.A is specified in the data sheets as "tRAS". Again this is a user-specified parameter.

Both of these are only a problem for short transfers, but this SDRAM controller automatically ensures that these requirements are met.

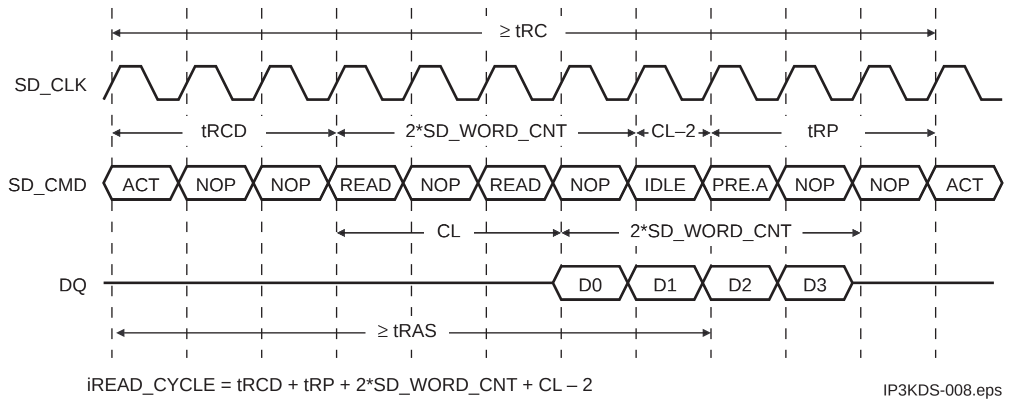


Figure 5-5 External SDRAM Read Protocol

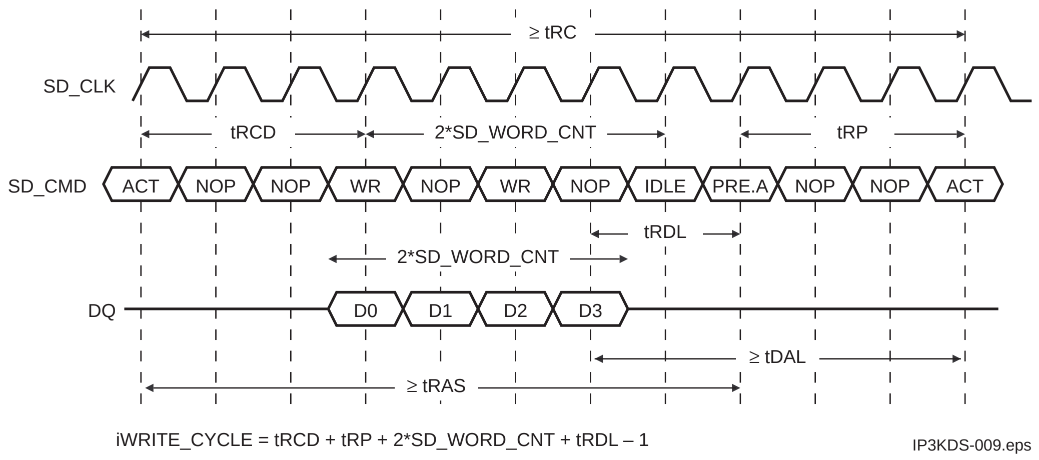


Figure 5-6 External SDRAM Write Protocol

For writes (see Figure 5-6), there are two additional parameters that also must be satisfied:

- The time from the last data written into the SDRAM before the PRE.A command can be initiated. This is specified in the user parameter tRDL.
- The time from the last data written into the SDRAM before the ACT command can be initiated: tDAL. This is not user selectable; instead, the previous parameter (tRDL) is programmed with a value to ensure that (tRDL + tRP ≥ tDAL) is met.

The primary commands used to access the external SDRAM are SD_READ and SD_WRITE. These commands are latched by the controller when it receives an SD_START pulse. The command is executed as soon as the SDRAM controller completes any task it is currently performing. Up to four outstanding user commands can be latched at any time; more than that will cause the FIFO to overflow and overwrite earlier entries. The hardware does not detect command queue overflows.

The controller makes use of the burst-length=2 mode of SDRAM components where the device automatically accesses two columns for every transaction. Every JEDEC compatible SDRAM device supports this mode. Thus, given a device size of 16-bits, the atomic transfer size supported by this controller is 32-bits or 1-word.

Addresses presented to the SDRAM controller must be word aligned. In-fact it is not possible to specify any non-aligned values, since address bits[1:0] are not available.

The required number of words to transfer is specified by the user in the SD_WORD_CNT parameter and the controller automatically terminates when a count of SD_WORD_CNT has been reached.

An SDRAM page is defined as all columns in the same bank and row. Software should take care when requesting data with a large SD_WORD_CNT near the end of an SDRAM page, because the controller will simply wrap around and continue to present data from the start of the same page. Different configurations of SDRAM with different capacities also have different page sizes, so it impossible to give a fixed guideline; Table 5-11 gives a general guideline.

Table 5-11 Address Alignment Guideline for Different Burst Lengths

SD_WORD_CNT	# Words	Burst-length (16-bit wide interface)
0000001	1	2 transfers
0000010	2	4 transfers
0000011	3	6 transfers
…	…	…
1111111	127	254 transfers
0000000	128	256 transfers

SDRAM provides a self-refresh mode in which the device refreshes itself without any outside intervention. While in self-refresh mode, SD_CKE is the only enabled input pin; all other inputs (including SD_CLK) are disabled and ignored. The SDRAM controller provides a method for the programmer to enter this mode using SD_SLEEP, while the SD_NOP command drives the signals needed to exit correctly from this mode. Note that there may be supplier specified minimum time constraints from self-refresh exit using an SD_NOP command before any new commands can be executed on the device; it is up to software to ensure that this condition is met.

Table 5-12 SDRAM Timing Parameters

Parameter	Description	Values	
SD_CL[1:0]	CAS Latency of the external SDRAM expressed in SD_CLK cycles.	This register must be programmed during initialization otherwise data transfer cannot occur. Behavior is undefined if modified during a data transfer. The value of CAS latency programmed into the internal SDRAM mode register is given in the following table:	
		SD_CL	**CAS Latency**
		01	1
		10	2
		11	3
		00	4
SD_RDL[1:0]	max(tRDL,(tDAL-tRP)) expressed in SD_CLK cycles	To ensure correct operation, this parameter should be set to the value of: int(max(tRDL,(tDAL-tRP))/SD_CLK + 1) i.e., the maximum value of tRDL or (tDAL-tRP) expressed in SD_CLK cycles with any fractions rounded up. This register must be programmed during initialization; otherwise data transfer cannot occur. Behavior is undefined if modified during a data transfer.	
SD_tRCD[2:0]	RAS to CAS Delay of the external SDRAM expressed in SD_CLK cycles.	To ensure correct operation, this parameter should be set to the value of: int(tRCD/SD_CLK + 1) i.e., fractions cause the value to be rounded up. This register must be programmed during initialization; otherwise data transfer cannot occur. Behavior is undefined if modified during a data transfer.	
SD_tRP[2:0]	Row Precharge time of the external SDRAM expressed in SD_CLK cycles.	To ensure correct operation, this parameter should be set to the value of: max(2,int(tRP/SD_CLK + 1)) i.e., the minimum supported value is 2; otherwise, fractions cause the value to be rounded up. This register must be programmed during initialization; otherwise data transfer cannot occur. Behavior is undefined if modified during a data transfer.	
SD_tRAS[3:0]	min time from ACT to PRE.A of the external SDRAM expressed in SD_CLK cycles	To ensure correct operation, this parameter should be set to the value of: int(tRAS/SD_CLK + 1) i.e., fractions cause the value to be rounded up. This register must be programmed during initialization; otherwise data transfer cannot occur. Behavior is undefined if modified during a data transfer.	

Table 5-12 SDRAM Timing Parameters

Parameter	Description	Values
SD_tRFC[3:0]	max(tRC,tRFC) expressed in SD_CLK cycles	To ensure correct operation, this parameter should be set to the value of: int(max(tRC,tRFC)/SD_CLK + 1) i.e., the maximum value of tRC or tRFC expressed in SD_CLK cycles with any fractions rounded up. This register must be programmed during initialization; otherwise data loss will occur. Behavior is undefined if modified during a data transfer. Programming Note: Not all device manufactures specify tRFC in their data sheet; in this case simply use the value for tRC. Examples: For the Micron MT48LC4M16A2 device, tRFC=66ns and tRC=60ns are specified for the “-7E” part; using an external SDRAM frequency of 62.5MHz, this parameter should be set with the value of 5. For the Samsung K4S641632D device, there is no separate tRFC specified – using the tRC value of 68ns for the “-70” part, again assuming that the external SDRAM interface is operating at a speed of 62.5MHz, the parameter should also be set to the value 5.
SD_EN_REFRESH	Enables the auto-refresh generation function of the SDRAM controller.	This must be disabled during SDRAM initialization. It may also be disabled while the SDRAM is in self-refresh mode ("SD_SLEEP") but must be enabled again on exit from that mode. If not disabled during self-refresh mode, the external SDRAM device will automatically ignore the refreshes that are sent to it. Programming Note: This bit must be set after the SD_REF[15:3] value is configured. The precise number of cycles to wait can be calculated by rounding up the result of: (SD_CLK ns / CORE_CLK ns) + 1. This allows the asynchronous boundary to be crossed safely.

Table 5-12 SDRAM Timing Parameters

Parameter	Description	Values
SD_REF[15:3]	Refresh period of the external SDRAM expressed in SD_CLK cycles	To ensure correct operation, this parameter should be set to the value of: int(refresh/SD_CLK) i.e., fractional values must always be rounded down. This register must be programmed during initialization; otherwise data loss will occur. Behavior is undefined if modified during a data transfer. This value is required because the SDRAM controller takes care of generating the auto-refresh commands internally. Any changes to this value take immediate effect while the SD_EN_REFRESH control is deasserted. Otherwise, changes take effect only when the refresh counts down to 1 and loads this value for the next cycle. Programming Note 1: This value is not always presented openly in a vendor's specs – e.g., specification might state: 64ms refresh period at 4K cycles. What this means is that this device requires 4K auto-refresh cycles in a 64ms period – i.e., an auto-refresh period of 15.625ms. To calculate the value required in this parameter, simply multiply the period by the SDRAM controller operating frequency – assuming 62.5MHz then this register would need to be set with the value 976. Programming Note 2: The result of the above calculation is assumed to fit within a 16-bit value SD_REF[15:0]; however the least significant 3-bits [2:0] are never set in the parameter. This is because the refresh sequence itself takes time to complete, and it is meaningless to request refreshes at a frequency faster than that. Care should be taken when writing the SD_REF parameter to program only the upper 13-bits and not to overwrite any previous state of SD_EN_REFRESH.

- -

5.5 Serializer/Deserializer (SerDes)

The IP3023 has two SerDes units, which support a variety of serial communication protocols, including GPSI, SPI, UART, USB, and 10Base-T Ethernet. By performing data serialization/deserialization in hardware, the CPU bandwidth needed to support serial communication is greatly reduced, especially at high baud rates. Providing two units allows easy implementation of protocol conversion or bridging functions between the two high-speed serial interfaces, such as a USB to 10Base-T Ethernet bridge.

One SerDes unit is associated with Port E, and one with Port F. Each SerDes unit uses up to 8 external digital signals shown in Table 5-16. Not all signals are used in all protocol modes. Refer to Table 5-15 for details on signal port pin usage in various protocol modes. In addition to the digital signals, there are also two analog signals used only in 10Base-T Ethernet mode: PFRDN and PFRDP. These analog signals are connected only to Port F; therefore, Port F has the only SerDes unit that can support 10Base-T Ethernet. The mapping of these signals onto the port pins is shown in Table 2-8 and Table 2-9.

SerDes Registers and Interrupts

Table 5-13 gives detailed descriptions of the registers and interrupts used by the SerDes units.

Table 5-13 SerDes Registers Interface

Field Name	Register	Bit(s)	Read/ Write	Description	
RX FIFO Overflow	Interrupt	13	RO	FIFO full during receive.	
RX FIFO Watermark	Interrupt	12	RO	Receive FIFO level greater than or equal to Receive FIFO watermark trigger level.	
RXERR	Interrupt	7	RO	10Base-T:	Asserted if a Manchester data phase error is detected.
				USB:	Asserted if SerDes has detected 7 consecutive ones.
RXEOP	Interrupt	6	RO	10Base-T and USB:	Asserted at end of packet.
				GPSI:	Asserted at deassertion of RxEn.
SYND	Interrupt	5	RO	10Base-T and USB:	Asserted when received data matches sync pattern (RSYNC).
TXBE	Interrupt	4	RO		Asserted when SerDes has consumed the data present at TXBUF[15:0]. The data may not have been completely transmitted across the output pins at the time this interrupt is asserted.
TXEOP	Interrupt	3	RO		Asserted when SerDes has completed transmitting all available data and no new data is available (as would be indicated with the assertion of TXBUF_VALID). In 10Base-T or USB modes, an EOP is transmitted after the last data.
SX LP	Interrupt	2	RO	10Base-T:	Asserted when a link pulse is detected.
			RO	USB:	Asserted when the bus goes idle after SerDes stops driving the bus. Indicates whether the SerDes is actively transmitting data.
RXBF	Interrupt	1	RO		Asserted when receive data is available.

Table 5-13 SerDes Registers Interface

Field Name	Register	Bit(s)	Read/ Write	Description	
RXXCRS	Interrupt	0	RO	10Base-T:	Asserted when carrier sense is gained or lost as configured through CRS_INT_POLARITY
				USB:	RxBusy is detected, indicating the SerDes is receiving data into the shift register (although there might not be any data in the data registers yet).
TXBUF_VALID	Set	16	WO		Writing a 1 here indicates that data in TXBUF is valid.
GLOBAL_EN	Control 0	31	R/W		Enables/disables SerDes pins without resetting the device. 0 = Disable SerDes output. SerDes I/O pins become accessible through GPIO. 1 = SerDes I/O pins are driven from SerDes as defined through the MODE register.
LOOP_BACK,	Control 0	30	R/W		Loop-back control. 0 = disable loopback 1 = enable loopback
TX_DATA_INV	Control 0	29	R/W		Invert all transmitted data.
TXSCNT[4:0]	Control 0	28:24	R/W		Transmit bit count.
MODE[7:0]	Control 0	23:16	R/W		SerDes mode/submode select. 23:20 PRS (Protocol Select) refer to Table 5-14. 19:18 SUBM (Submode); depends on the protocol selected by PRS. 17:16 unused
CLKDIV[15:0]	Control 0	15:0	R/W		Clock divider for generating serial I/O clock from I/O clock.
SYNCMASK[7:0]	Control 1	31:24	R/W	USB :	Sync mask for RSYNC.
				10BaseT:	31 reserved 30:27 Configure preamble count 0: 24 pairs 1: 2: 20 pairs 3: 4: 16 pairs 5: 4 pairs 6: reserved 7: reserved 26 reserved 25 0: sync detected if 6 “10” pairs + “11” is received 1: use preamble count (syncmask[6:3]) for number of “10” used for sync detection. 24 reserved

Table 5-13 SerDes Registers Interface

Field Name	Register	Bit(s)	Read/ Write	Description
RSYNC[7:0]	Control 1	23:16	R/W	USB: Synchronization pattern RSYNC[7:0]. 10Base-T: 23:18 reserved 17 SQUELCHEN 0 = Squelch disabled 1 = Squelch enabled 16 DRIBBITEN 0 = Hardware handles dribble bit 1 = Software handles dribble bit
BIT_ORDER	Control 1	9	R/W	Controls serialization order: 0 = LSB first 1 = MSB first
CRS_INT_POLARITY	Control 1	8	R/W	10Base-T: 0 = RXXCRS interrupt on gain of carrier 1 = RXXCRS interrupt on loss of carrier
SPI_MASTER_SEL	Control 1	7	R/W	SPI, GPSI: 0 = Master 1 = Slave
USB_SYNC_IGNORE	Control 1	6	R/W	USB : 1 = Do not detect sync.
REV_POLARITY_EN	Control 1	5	R/W	Invert all received data.
RXSCNT[4:0]	Control 1	4:0	R/W	Receive count interrupt level.
SQUELCH TRIM [7:0]	Control 2	23:16	R/W	Squelch trim value; set to 0xFF when SerDes 10Base-T is selected.
TXBUF[15:0]	Control 2	15:0	R/W	Data for transmit operations.
LINK_POLARITY	Status 0	9	RO	10Base-T: Link pulse detected with reverse polarity.
CRS_STATUS	Status 0	8	RO	10Base-T only: Current state of carrier.
RXCTR[4:0]	Status 0	4:0	RO	Shows actual number of received bits. Exceptions for the last transfer are: • Shows number of bits if less than 8 • Shows 8 if less than 16 and greater than 8 • Shows 16 if received count is ≥ 16 and RXSCNT = 16
RX DATA [15:0]	RX FIFO	15:0	RO	Received data

5.5.1 SerDes TX/RX Buffers

RX FIFO

Received data is placed into the port's RX FIFO. The SerDes asserts the RXBF interrupt to indicate when new data is available from the FIFO (the interrupt mask bit must be set to cause an interrupt). The RX FIFO Overflow Interrupt indicates when the receive FIFO becomes full during receive, and the RX FIFO Watermark Interrupt indicates when the receive FIFO level exceeds the receive FIFO watermark trigger level.

The TXBUF Register Field

The 16-bit TXBUF field of the Function Control 2 register is for loading data to be transmitted. Asserting TXBUF_VALID in the Interrupt Set register signals that the data in TXBUF is ready to be transmitted. The SerDes asserts the TXBE interrupt to indicate when the data has been transmitted and the register is ready to be loaded with new data (the interrupt mask bit must be set to cause an interrupt).

5.5.2 SerDes Configuration

Software prepares a SerDes unit to receive data by programming the receive shift count register field (RXSCNT) and the clock divider (CLKDIV) appropriately for the selected protocol. The RXSCNT is copied to an internal counter, and when that number of bits of data has been received, the received data is loaded into the RX FIFO.

In 10Base-T, GPSI, or USB mode, when an EOP is detected, the RXCTR register field is loaded with the number of bits actually received (with the exception of the last transfer), the RXEOP interrupt is asserted, and the data bits are loaded into the RX FIFO (the interrupt mask bit must be set to cause an interrupt).

The TXP and TXM signals correspond to the differential outputs of the USB or Ethernet bus. Other serial protocols require only one output pin, which is TXP by default.

The TXP and TXM pins of Port F have high current outputs for driving Ethernet magnetics directly without the use of transceivers.

In 10Base-T mode, the transmit pre-emphasis requirement enables the TXPE and TXME outputs, which have a 50ns-delayed version of the transmit output that is resistively combined outside the chip before driving the magnetics.

For transmitting, software must specify the number of bits to transmit (in the TXSCNT register field), load the data into the TXBUF register, and assert TXBUF_VALID to signal availability of new data. This data is then transferred to an internal register, from which it is serially shifted out to the transmit logic. The TXBE interrupt is asserted when the data has been transferred from the TXBUF register (the interrupt mask bit must be set to cause an interrupt).

When there is a transmit buffer underrun event (i.e. all of the data has been shifted out from the internal register, but the TXBUF register has not been reloaded), an EOP condition is generated on the TXP and TXM outputs after an internal counter decrements to zero. The TXEOP interrupt is asserted when an underrun event occurs (the interrupt mask bit must be set to cause an interrupt).

For protocols other than USB and Ethernet, the EOP generator is bypassed.

5.5.3 Protocol Modes

Table 5-14 shows the protocols selected by the PRS bits in the MODE field of the Function Control 0 register. The selection of protocol affects which registers and register fields are used, for example the RSYNC field of the Function Control 1 register is only used in the USB and 10Base-T modes. The protocol mode also affects the signal usage, as shown in Table 5-15. Pins not used for protocols can be used for general I/O. Table 5-16 provides more information about each signal.

Table 5-14 Protocol Selection

PRS	Mode
0001	10Base-T
0010	USB Bus
0011	UART
0101	SPI
0110	GPSI

Table 5-15 SerDes Protocol Modes And Signal Usage

Signal	Port E or Port F Pin	SerDes Mode						
		10Base-T Ethernet (Port F Only)	USB Bus	UART	SPI		GPSI	
					Master	Slave	Master	Slave
RXD	0		RCV (I)	RXD	DI (I)	DI (I)	RxD (I)	RxD (I)
RXM	1		VM (I)					
RXP	2		VP (I)			SS (I)	RxEN (I)	RxEN (I)
CLK	3		Optional		SCLK	SCLK		RxCLK (I)
TXME	4	TxD– (O)						TxBUSY (I)
TXM	5	Tx– (O)	VMO (O)				TxCLK/ RxCLK (O)	TxCLK (I)
TXP	6	Tx+ (O)	VPO (O)	TXD	DO (O)	DO (O)	TxD (O)	TxD (O)
TXPE	7	TxD+ (O)	OE (O)				TxEN (O)	TxEN (O)
PFRDN		RX– (I)						
PFRDP		RX+ (I)						
Any GPIO pin							TxBUSY (O)	
Any GPIO pin							COLLISION (O)	COLLISION (I)

I: Input
O: Output

Table 5-16 SerDes Signals

Pin	Description
RXD	Serial data for USB, UART, SPI and GPSI modes.
RXM	Negative-side differential input (USB only).
RXP	Positive-side differential input (USB only), Slave Select (for SPI Slave), or data valid (GPSI).
CLK	Serial Clock in SPI or GPSI Slave modes, optional external serial I/O clock input for USB or UART modes.
TXME	Negative-side delayed differential output for pre-emphasis (10base-T Ethernet), or TxBUSY in GPSI mode.
TXM	Negative-side differential output (10base-T Ethernet and USB modes), transmit clock (GPSI Slave), or transmit and receive clock (GPSI Master).
TXP	Positive-side differential output (10base-T Ethernet and USB modes), or serial data (UART, SPI and GPSI modes).
TXPE	Positive-side delayed differential output for pre-emphasis (10base-T Ethernet), output enable for external transceiver (USB), or data valid for GPSI mode.
PFRDN	Negative-side analog differential input, used for 10base-T Ethernet squelch function (port F only).
PFRDP	Positive-side analog differential input, used for 10base-T Ethernet squelch function (port F only).

5.5.4 10base-T Ethernet

Table 5-17 10base-T Ethernet Interface Signal and Port Pin Usage

10base-T Signal Name	SerDes Signal Name	Port F Pin Location	Direction	Description
Tx+	TXP	6	Output	Plus-side differential output
Tx-	TXM	5	Output	Minus-side differential output
TxD+	TXPE	7	Output	Plus-side differential output with pre-emphasis
TxD–	TXME	4	Output	Minus-side differential output with pre-emphasis
Rx+	PFRDP		Input	Plus-side analog differential input, used for 10base-T Ethernet squelch function
Rx–	PFRDN		Input	Minus-side analog differential input, used for 10base-T Ethernet squelch function

Hardware

Only Port F supports 10Base-T Ethernet. Table 5-17 shows signal and port pin usage for 10Base-T Ethernet on Port F. Port pins not used by 10Base-T Ethernet are available for other functions.

Figure 5-7 shows the clock/data separation and End-of-Packet (EOP) detection logic of a 10Base-T receiver unit. The PFRDP and PFRDN pins correspond to the differential inputs. Providing both inputs allows sensing of an EOP condition. To set up a SerDes unit for 10Base-T Ethernet, the input data from a differential line receiver is connected to the RX+ and RX– input. The signals designated Tx+, Tx–, TxD+, and TxD– correspond to the TXP, TXM, TXPE, and TXME pins of the corresponding SerDes. These pins are connected to an RJ45 jack through a transformer with terminations.

Figure 5-8 shows an example circuit. Rtxpe, Rtxme, Rtxp, Rtxm and RL values vary depending on the Ethernet magnetics used. Refer to the application note “SerDes Based Native Ethernet” for more details.

For 10Base-T Ethernet operation, Port F SerDes is equipped with a squelch circuit for discriminating between noise, link pulses, and data. Link pulses are sent periodically to keep the channel open when no data is being transmitted. The squelch circuit handles link pulse detection, link pulse polarity detection, carrier sense, and EOP detection.

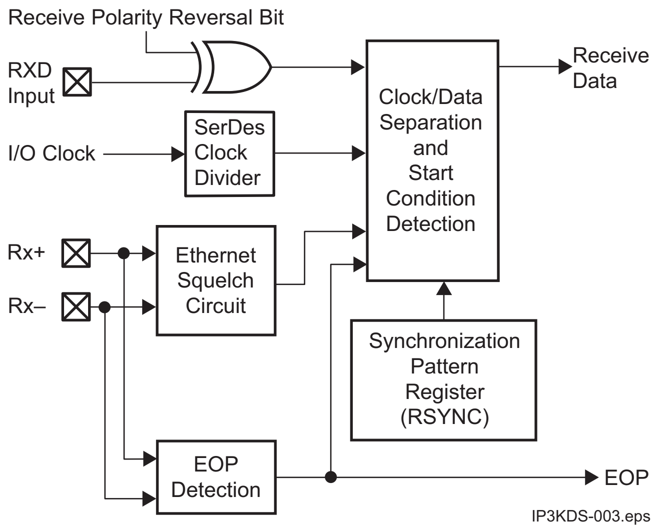

Receive Polarity Reversal Bit
RXD Input
I/O Clock
SerDes Clock Divider
Clock/Data Separation and Start Condition Detection
Receive Data
Rx+
Rx–
Ethernet Squelch Circuit
Synchronization Pattern Register (RSYNC)
EOP Detection
EOP
IP3KDS-003.eps

Figure 5-7 Clock/Data Separation and EOP Detection

The 10Base-T mode requires only a fixed SFD (start of frame) pattern, so the SFD pattern for 10Base-T is hard-wired to be 11010101 and the RSYNC field of the Function Control 1 register is used to configure features of 10Base-T other than SFD pattern. The incoming data stream, after passing through the polarity inversion logic (which can be turned on or off with REV_POLARITY_EN) is compared to the synchronization pattern. Once a match is found, an internal counter is set to zero and data is shifted into a shift register. The synchronization matching operation is then disabled until an EOP condition is detected, because the synchronization pattern might also be embedded in the data stream.

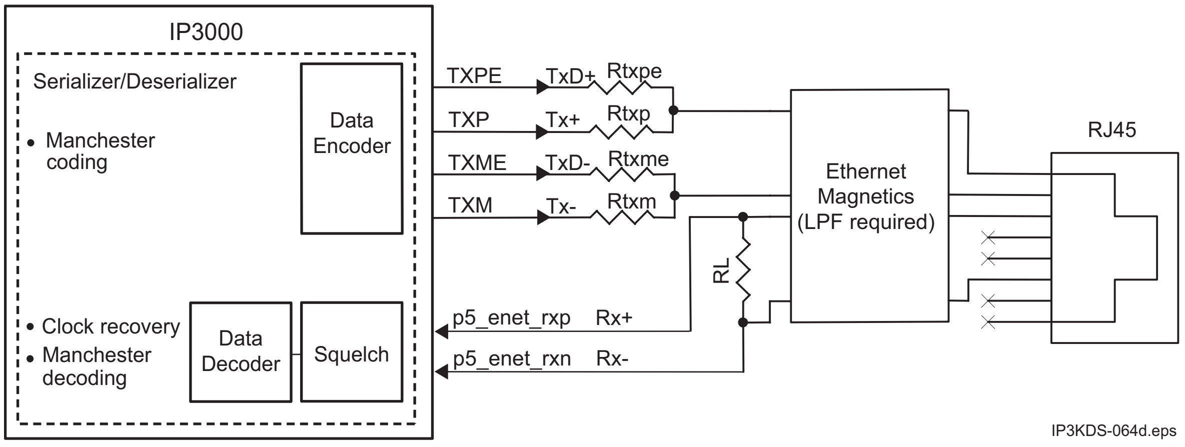


Figure 5-8 Ethernet Interface Example

Figure 5-9 shows the receive data paths. When an EOP is detected the RXCTR register field is loaded with the number of bits actually received, the RXEOP interrupt is asserted, and the data bits are loaded into the RX FIFO (the interrupt mask bit must be set to cause an interrupt).

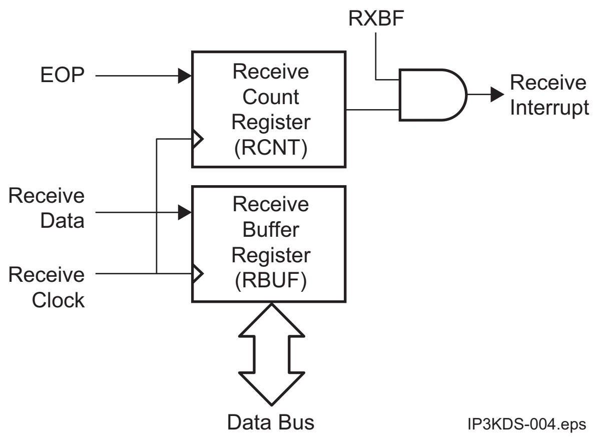


Figure 5-9 Receive Data Paths

The data encode block performs 10Base-T Manchester encoding. The encoded TX signal is sent to the TX pins in a differential mode. The encode block is bypassed for all other protocols. The TXP and TXM pins have high current outputs for driving Ethernet magnetics directly without the use of transceivers. The pre-emphasis TX outputs are enabled on TXPE and TXME outputs, which have a 50ns-delayed and inverted version of the transmit outputs. The resistively combined TX outputs outside the chip are used to drive the magnetics. The output pins of the serializer are driven low when not transmitting.

Figure 5-10 shows the transmit data paths. For transmitting, software must specify the number of bits to transmit (specified in the TXSCNT register field), load the data in the TXBUF field of Function Control 2 register, and assert TXBUF_VALID in the Interrupt Set register. This data is then transferred to an internal register, from which it is serially shifted out to the transmit logic. The TXBE interrupt is asserted when the data has been transferred from the TXBUF internal register (the interrupt mask bit must be set to cause an interrupt). When there is a transmit buffer underrun event (i.e., all of the data has been shifted out from the internal register, but TXBUF has not been reloaded), an EOP condition is automatically generated on the TX output pins after an internal counter decrements to zero. The TXEOP interrupt is asserted when an underrun event occurs (the interrupt mask bit must be set to cause an interrupt).

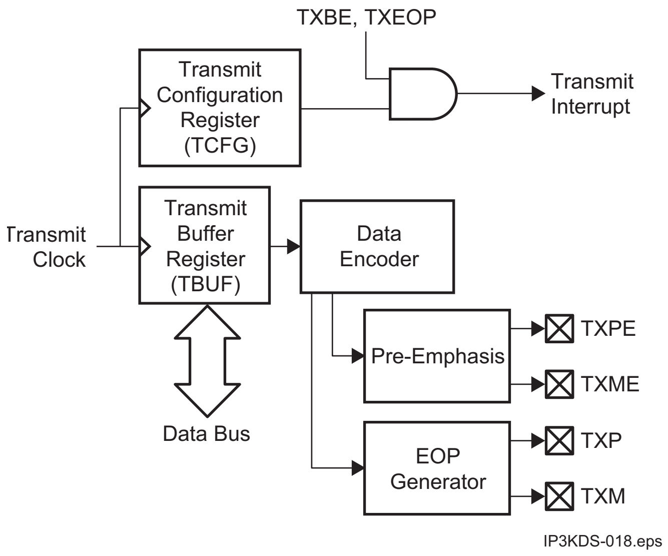


Figure 5-10 Transmit Data Paths

Software

The SerDes 10Base-T mode is designed to run at a fixed 8x oversampling for line receiver. Therefore, a fixed 80MHz master must be configured through the CLKDIV field of the Function Control 0 register. The Serial I/O PLL clock multiplier must be programmed to be an integer multiple of 80MHz for 10Base-T operation. The received data stream is used, together with the clock recovery circuit, to recover the original transmit clock and data.

Software must perform the following functions:

- Polarity detection and reversal.
- Carrier sense.
- Jabber detection.
- Link integrity test and link pulse generation.
- Random back off in case of collision.
- When a collision is detected, sending a 32-bit jam sequence. Collisions can be detected by positive detection of carrier sense during active transmission.
- Formation of Ethernet packet by putting the preamble, SFD, destination address, source address, length/type, and MAC client data into the transmit buffer. Frame check computation can be done in software or through the CRCGEN instruction.
- MAC layer functions.

5.5.5 USB

Table 5-18 USB Interface Signal Usage

USB Signal Name	SerDes Signal Name	Port E and F Pin	Direction	Description
VP	RXP	2	Input	Plus-side differential input
VM	RXM	1	Input	Minus-side differential input
VPO	TXP	6	Output	Plus-side differential output
VMO	TXM	5	Output	Minus-side differential output
OE	TXPE	7	Output	Output enable
RCV	RXD	0	Input	Receive data
Clock	CLK	3	Input	External clock input (optional)

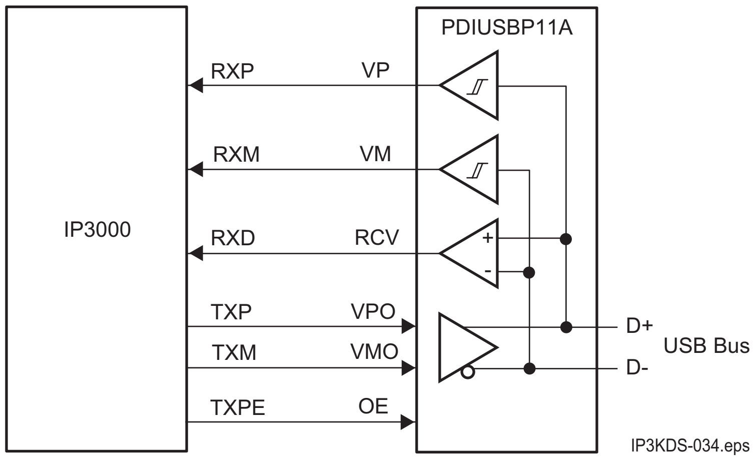


Figure 5-11 USB Interface Example

Each SerDes provides support for USB revision 1.1 host and device modes of operation.

Hardware

To set up a SerDes unit for USB mode, the received data output of the USB transceiver should be connected to RXD. The VP and VM pins of the transceiver are connected to the RXP and RXM pins to allow detection of the EOP condition. Figure 5-11 shows the connections required between an external USB transceiver and the IP3023. Table 5-18 shows the mapping of USB signals to the SerDes pins. For additional hardware configuration information, USB reference hardware designs may be available on the Ubicom technical support portal for registered development kit users. Please visit the portal for the latest information, or contact Ubicom.

Software

The MODE register field must be programmed with values for the desired USB mode, Full Speed or Low Speed. The serial I/O clock divider CLKDIV also needs to be programmed to generate the appropriate frequency according to the USB submode selection. Table 5-19 shows the serial I/O PLL clock frequencies required for the low and full speed modes of the USB. For example, if the clock from the serial I/O PLL into the SerDes is 240 MHz, it can be programmed at 48 MHz for full speed with a divisor of 5 (encoded as 4). A divisor of 40 (encoded as 39) is required for low-speed USB. Table 5-20 shows the submode values for selecting the low- or high-speed modes.

Table 5-19 Required Clock Frequencies from Serial I/O Clock in USB Mode

Protocol	Receive
USB 2.0 Full Speed *	48 MHz
USB 2.0 Low Speed	6 MHz

* On-chip USB is 2.0 compatible, supporting maximum data rates up to 12Mbps, but not high-speed mode (480Mbps).

Table 5-20 Submodes for USB

Name	Description
SUBM1:0	Submode select for USB mode: 01 = Low-speed USB interface 10 = High-speed USB interface

In USB mode, the SerDes uses two registers, RSYNC and SYNCMASK, to detect the sync pattern marking the beginning of a USB data stream. This sequence is defined to be “7 zeros and a single 1” by the USB specification, and only the last 3 bits need to be matched to start receiving data, also defined in the specification. In order to achieve this, RSYNC must be programmed with 0x80 and SYNCSMASK must be programmed with 0xE0.

Receive behavior is controlled by several register fields. For normal USB operation, USB_SYNC_IGNORE should be 0, REV_POLARITY_EN should be 0, and RXSCNT should be set to the desired number of bits received, usually 8 or 16. BIT_ORDER should be cleared to make sure receive is performed LSB first. Once the SerDes matches the USB SYNC pattern, the internal receive count is reset to zero and the SerDes receives bits from the line until either the desired count is received or an EOP is encountered, at which point the received data is transferred to the RX FIFO. If more data is coming in, the procedure will be repeated. Software is responsible for reading the data from the RX FIFO fast enough to avoid overflow. When the EOP is received, the SerDes remains idle until the next match of the SYNC pattern.

Transmit behavior is controlled by several register fields. For normal USB operation, LOOP_BACK should be 0, TX_DATA_INV should be 0, and TXSCNT bits should be set to the number of bits to transmit. Transmit is initiated by writing the data to the TXBUF register, then asserting TXBUF_VALID. If the transmit count needs to be changed, it must be changed before setting TXBUF_VALID. For continued transmission, the TXBUF register has to be written before the TXSCNT count is reached. Otherwise, the SerDes automatically inserts the EOP signaling.

While receiving data, the clock/data separation circuit performs NRZI decoding, after which bit unstuffing is performed. This means every bit after a series of six consecutive ones is dropped. On transmit, the SerDes performs bit stuffing, and the clock/data separation circuit NRZI encodes the data.

Note: While configured for USB mode, the SerDes cannot be configured to interrupt on carrier status (RXXCRS).

Software must perform the following functions to implement the USB protocol for a device:

- CRC generation and checking (can be done with the CRCGEN instruction).
- Detecting reset of the device function, which is indicated by 10 milliseconds of a single-ended zero (SE0) condition on the bus.
- Detecting the suspend state, which is indicated by more than 3 milliseconds of idle. Software must make sure that the suspend current of 500 µA will be drawn after 10 milliseconds of bus inactivity.
- Formation of the USB packet by putting the sync, pid, and data into the transmit data registers and setting the proper count.
- Endpoint and device management and other higher level protocol tasks.

Timing Considerations

USB relies on certain timing limitations for error detection and recovery. Response time requirements are specifically harder to meet. The ISR for USB must be carefully structured to satisfy these requirements, and this is possible because of IP3023's deterministic ISR execution times. The time from the SE0 on bus to the RXEOP indication is about 208 ns. The time from writing to TX data registers and the data put on the bus is about 125 ns. Software tasks like address, error, CRC checking, and determining the endpoint response must be carefully timed and cycle counted to assure that the required timing limitations are satisfied.

5.5.6 UART

For UART operation, two internal divide-by-16 circuits are used. The receive section and the transmit section use two divided-by-16 clocks that potentially can be out of phase. This is due to the nature of the UART bus transfers. The receive logic, based on the 16x bit clock, samples the incoming data for a falling edge. Once the edge is detected, the receive logic counts 8 clock cycles and samples the number of bits specified in the RXSCNT register using the bit clock (which is obtained by dividing the clock source by 16).

Hardware

Figure 5-12 shows an example circuit to connect the SerDes in UART mode. Table 5-21 shows the UART signal to port pin usage.

Software

To set up a SerDes unit for UART mode, select UART mode in the PRS bits of the MODE register field. This causes the data to be clocked in after a valid start bit is detected. Make sure that the polarity selected by the REV_POLARITY_EN and TX_DATA_INV matches the polarity provided by the RS-232 transceiver (most of them are inverted). Make sure the BIT_ORDER is compatible with the data format (RS-232 uses LSB-first bit order). The receiver uses 16X oversampling, so select a serial I/O clock divisor (CLKDIV) that is 16 times the desired baud rate.

To operate in UART mode, depending on the application, either transmit or receive can be performed first. In both cases, the shift count register must be programmed with a bit count that is appropriate for the format. The bit count depends on the number of data bits, stop bits, and parity bits. The start bit is included in the bit count. The receiver does not check for the presence of stop bits. To detect framing errors caused by missing stop bits, increase the receiver's bit count (i.e., the RXSCNT field) and test the trailing bit(s) in software.

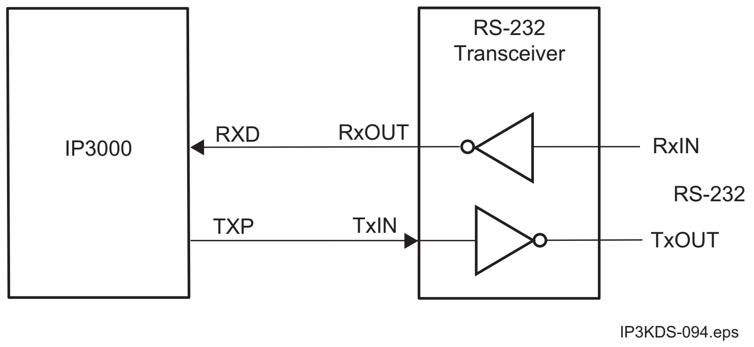


Figure 5-12 UART Interface Example

Table 5-21 UART Interface Signal Usage

UART Signal Name	SerDes Signal Name	Port E and F Pin	Direction	Description
RXD	RXD	0	Input	Receive data
TXD	TXP	6	Output	Transmit data

5.5.7 SPI

Hardware

Figure 5-13 shows example circuits to connect the SerDes in SPI mode. Table 5-23 and Table 5-24 show the SPI signal to port pin usage. Refer to Table 5-13 for the SerDes port register interface.

Configuration

The SerDes can be configured for either master or slave mode:

SPI_MASTER_SEL = 1: slave
SPI_MASTER_SEL = 0: master

The SerDes SCK idle-level (i.e., when $\overline{SS}$ is deasserted) can be configured by the SUBM field of MODE (refer to Table 5-25).

MODE[3] (CPOL) = 0: idle is low
MODE[3] (CPOL) = 1: idle is high

Finally, the SerDes can be configured for the phase relationship of the SDO/SDI pins with respect to the SCK edge:

MODE[2] (CPHA) = 0:
  SDO is set up by the other device a half clock period before the first edge following the assertion of $\overline{SS}$. Therefore, SDI will be sampled by this device (the slave) on the first edge (transition).

MODE[2] (CPHA) = 1:
  SDO is set up by the other device on the first edge following the assertion of $\overline{SS}$. Therefore SDI will be sampled by this device (the slave) on the second edge (transition).

Note: The use of the term “edge” in the above paragraphs implies any transition, not a specific type of transition (i.e., rising or falling). Therefore, “first edge” implies a rising edge when CPOL=0, and implies a falling edge when CPOL=1.

In the SPI scheme implemented by Motorola, which the IP3023 follows, data being emitted on SDO and data being sampled on SDI always occur on opposing edges of the clock, on either master or slave. Transmitting and sampling on the same edge of the clock is not supported by the SerDes.

CPOL, in conjunction with CPHA, determines which clock edges the SerDes will be using to output and sample data on, as given by Table 5-22.

Table 5-22 SerDes Output and Sample Configuration

CPOL	CPHA	
0	0	output on falling, sample on rising
0	1	output on rising, sample on falling
1	0	output on rising, sample on falling
1	1	output on falling, sample on rising

When the SerDes is configured as a slave, the state of the SDO line when $\overline{SS}$ is deasserted is determined by the value in the RxOUT GPIO register for that pin, which the user can configure.

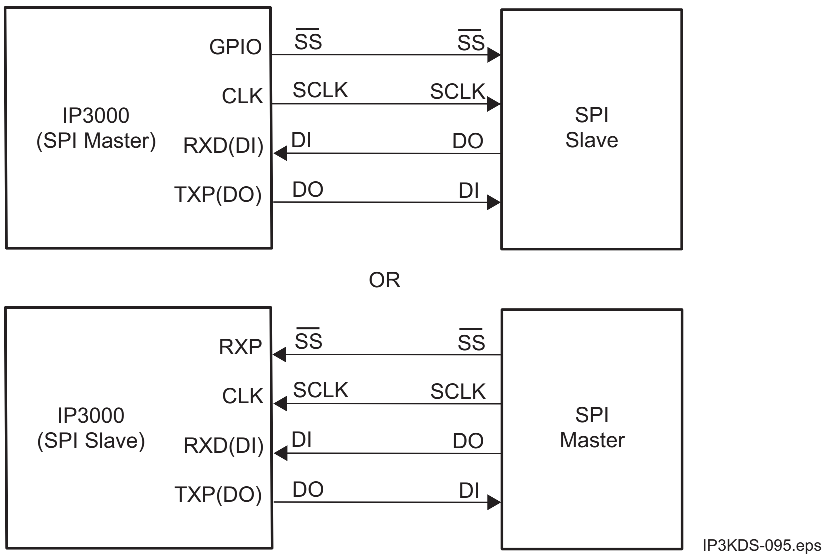


Figure 5-13 SPI Interface Examples

Table 5-23 SPI Master Interface Signal Usage

SPI Device Signal Name	IP3023 SPI Signal Name	SerDes Signal Name	Port E and F Pin	Direction	Description
SCLK	SCLK	CLK	3	Output	Serial clock output in master mode, input in slave mode
DO	DI	RXD	0	Input	Receive data
DI	DO	TXP	6	Output	Transmit data
SS	SS	GPIO	any other	Output	Slave select pin used in slave mode only (Master select handled by software)

Table 5-24 SPI Slave Signal Usage

SPI Device Signal Name	IP3023 SPI SIgnal Name	SerDes Signal Name	Port E and F Pin	Direction	Description
SCLK	SCLK	CLK	3	Input	Serial clock output in master mode, input in slave mode
DO	DI	RXD	0	Input	Receive data
DI	DO	TXP	6	Output	Transmit data
SS	SS	RXP	2	Input	Slave select pin used in slave mode only (Master select handled by software)

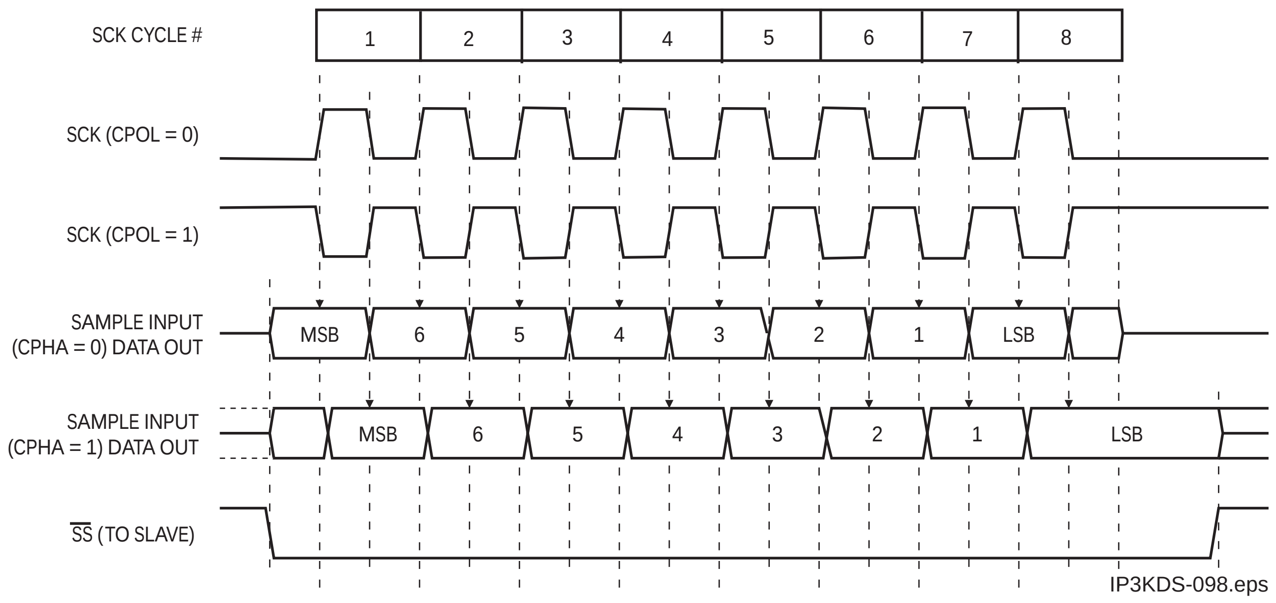


Figure 5-14 SPI Signal Timing

Table 5-25 Submodes for SPI

Name	Description
SUBM1:0	Submode select for SPI mode 00 = Positive clock polarity, receive on rising edge, transmit on falling edge 01 = Positive clock polarity, receive on falling edge, transmit on rising edge 10 = Negative clock polarity, receive on falling edge, transmit on rising edge 11 = Negative clock polarity, receive on rising edge, transmit on falling edge

5.5.8 SerDes-Based GPSI

Hardware

Figure 5-15 shows example circuits to connect the SerDes in GPSI (General Purpose Serial Interface) mode. Table 5-26 shows the GPSI signal to port pin mapping in Master mode, and Table 5-27 shows the GPSI signal to port pin mapping in Slave mode.

Software

GPSI is a general-purpose, point-to-point, full-duplex serial bus protocol. Only two devices are allowed to exist on a bus. The GPSI PHY device is responsible for maintaining bus timing by driving two continuously running clocks, TxClk and RxClk. The device that does not drive the clocks is the MAC device. The TxEn and TxD signals are synchronized to the TxClk clock. The RxD and RxEn signals are synchronized to the RxClk clock.

The COLLISION and TxBUSY signals do not participate in actual data transfer on the GPSI bus. COLLISION and TxBUSY provide additional flow control capabilities for the software device driver. The COLLISION signal indicates that a PHY device has detected a collision condition. This signal is useful only when the SerDes is connected to a PHY device or acting as a PHY device.

The TxBUSY signal is used by a GPSI device to indicate that the device is currently busy, and that another device should not attempt to start a data transfer.

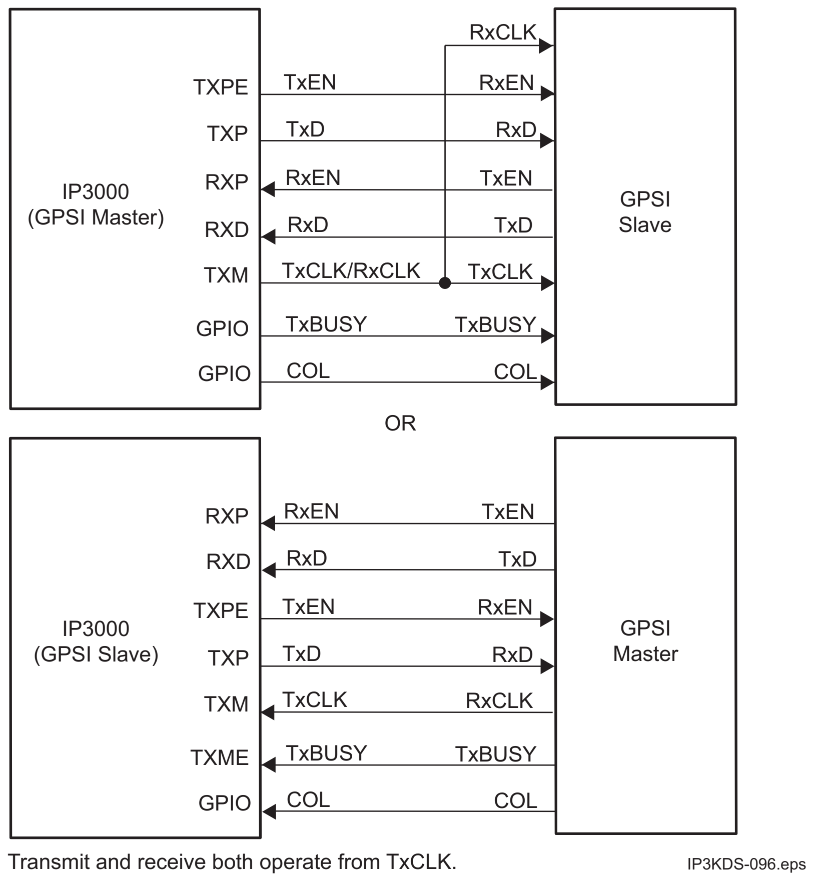


Figure 5-15 GPSI Interface Examples

Table 5-26 IP3023 GPSI Master Interface Signal Usage

GPSI Slave Signal Name	IP3023 GPSI Signal Name	SerDes Signal Name	Port E and F Pin	IP3023's Direction	Description
TxCLK and RxCLK	TxCLK and RxCLK	TXM	5	Output	Transmit and Receive clock
TxD	RxD	RXD	0	Input	Transmit data
TxEN	RxEN	RXP	2	Input	Transmit data valid
RxD	TxD	TXP	6	Output	Receive data
RxEN	TxEN	TXPE	7	Output	Receive data valid
TxBUSY	TxBUSY	GPIO	-	Output	Indicates a data transfer in progress (handled by software)
COLLISION	COLLISION	GPIO	-	Output	Indicates a collision at PHY layer (handled by software)

Note: In GPSI master mode, the TXM SerDes pin should be used by the GPSI slave for both TxCLK and RxCLK inputs.

Table 5-27 IP3023 GPSI Slave Interface Signal Usage

GPSI Master Signal Name	IP3023 GPSI Signal Name	SerDes Signal Name	Port E and F Pin	IP3023's Direction	Description
RxCLK	TxCLK	TXM	5	Input	Transmit clock
RxD	TxD	TXP	6	Output	Transmit data
RxEN	TxEN	TXPE	7	Output	Transmit data valid
TxCLK	RxCLK	CLK	3	Input	Receive clock
TxD	RxD	RXD	0	Input	Receive data
TxEN	RxEN	RXP	2	Input	Receive data valid
TxBUSY	TxBUSY	TXME	4	Input	Indicates a data transfer in progress (handled by software)
COLLISION	COLLISION	GPIO	-	Input	Indicates a collision at PHY layer (handled by software)

Table 5-28 Submodes for GPSI

Name	Description
SUBM1:0	Submode select for GPSI mode 00 = Receive on rising edge, transmit on falling edge 01 = Receive on falling edge, transmit on falling edge 10 = Receive on rising edge, transmit on rising edge 11 = Receive on falling edge, transmit on rising edge

5.6 Media Independent Interface (MII)

The IP3023 includes four MII controllers to support signaling and data transfer as defined in IEEE 802.3. This includes data transmit, data receive, and carrier and collision detect. The MII controllers, however, do not support the station management functions of MII (no MDIO/MDC signaling). Additionally, these controllers implement in hardware most of the functions of an Ethernet MAC, including packet filtering based on frame size, and receive CRC checking. Station management is handled through a different function/port.

Each MII controller uses its copy of the signals shown in Table 5-29. For the mappings of these signals to I/O pins refer to the appropriate port signal map in Section 2.3. Each MII controller has its copy of the port registers. Table 5-30 gives a detailed description of the port registers used for MII.

MII controllers are available at each of Ports C, D, and H, and one MII controller is available at the combined Port E and Port F, using some pins from each port. Both Port E and Port F must select the MII function, but otherwise only the registers of Port E manage the MII controller.

Table 5-29 MII Signals

Signal Name	Port C Pins	Port D Pins	Port E/ F Pins		Port H Pins	I/O	Description
			Port E	Port F			
CRS	0	0	0		0	I	Carrier Sense
COL	1	1	1		1	I	Collision
TXD [3:0]	2:5	2:5	2:5		2:5	O	Transmit Data
TX_EN	6	6	6		6	O	Transmit Enable
TX_CLK	7	7	7		7	I	Transmit Clock
TX_ER	8	8		0	8	O	Transmit Error
RX_ER	9	9		1	9	I	Receive Error
RX_CLK	10	10		2	10	I	Receive Clock
RX_DV	11	11		3	11	I	Receive Data Valid
RXD [3:0]	15:12	15:12		7:4	15:12	I	Receive Data

Table 5-30 MII Register Interface

Field Name	Register		Description
THRESHOLD_INT	Interrupt		The frame currently being received has reached the threshold at which it will not be discarded (32 bytes).
RX_EOP_INT	Interrupt		End-of-packet detected during receive
RX_SFD_INT	Interrupt		Start of Frame delimiter detected during receive
RX_ERR_INT	Interrupt		Error detected during receive
TX_EOP_INT	Interrupt		End of transmission
COL_INT	Interrupt		Collision detected during transmission
CRS_INT	Interrupt		Carrier sense
ODD_NIB_ERR_INT	Interrupt		Odd nibble reception error
FALSE_CARRIER_INT	Interrupt		False carrier message
TX_START	Set		Writing a 1 to this bit starts transmission.
CLK_DIV[7:0]	Control 0		Clock divisor for physical-side mode
REVERSE_MII	Control 0		Set the MII controller into physical-side mode.

Table 5-30 MII Register Interface

Field Name	Register		Description
HALF_DUPLEX	Control 0		Set the MII controller into half duplex mode. Enabling half duplex mode causes the MII controller to automatically initiate a JAM sequence when collision is detected.
RX_EN	Control 0		Enable the receiver. Until the receiver has been enabled, the MII interface ignores all incoming traffic.
TX_BYTE_COUNT[15:0]	Control 0		Number of bytes of data to be transmitted. The count includes the preamble, SFD, and CRC, as well as the data payload. A value of 0 results in undefined behavior.
CRC_OK	Status 0		A 1 at this location indicates that the CRC sent with the most recently received packet matches the CRC calculated on the payload of the same packet. The CRC is only calculated over whole bytes; therefore, an odd nibble at the end of a frame is not included in any CRC calculation.
RX_FIFO_SELECT	Status 0		Indicates to which FIFO the currently received frame is being written. 0 indicates FIFO 0 1 indicates FIFO 1
COLLISION	Status 0		This bit reflects the state of the COL signal after being synchronized to the core clock domain.
CARRIER_SENSE	Status 0		This bit reflects the state of the CRS signal after being synchronized to the core clock domain.
RX_BYTE_COUNT[15:0]	Status 0		Total number of bytes received, including the CRC, but not including the SFD or any part of the preamble. This field is latched at the end of a packet and maintains the receive byte count value up to the reception of the next frame which reaches or exceeds the receive packet threshold (32 bytes).
RX BYTE[4:0]	RX FIFO		RX_BYTE[0] or TX_BYTE[0]) is the most significant byte; bit 0 is the least significant bit. When transferring a number of bytes that is not a multiple of 4, the valid data is placed in the low order bytes. Inclusion of undefined bytes in a 32-bit word is permissible only in the last word of a packet.
TX BYTE[4:0]	TX FIFO		

5.6.1 Receive Sequence

The MII receiver moves data from the MII interface to the receive FIFO and signals exceptional conditions.

For both normal data reception and data reception with errors, data is moved into the receive FIFO for access by software. In the case of data reception with errors, a processor interrupt is generated. For normal interframe as well as for reserved messages, the MII controller takes no action. For a false carrier message, the MII receiver asserts the FALSE_CARRIER_INT interrupt.

5.6.1.1 Receive Start

1. The RX_EN bit must be set for the MII receiver to become active. If RX_EN is set while the RX_DV signal is active, the MII receiver waits until RX_DV is deasserted before becoming active.
2. Receiver activity is independent of the state of CRS (carrier sense).
3. Upon detection of an SFD, the RX_SFD_INT is asserted.

4. At reception of the minimum frame length (32) number of bytes,
   - The RX_PKT_SIZE_THRESHOLD interrupt is asserted.
   - The RX_FIFO_SELECT bit changes to indicate which receive FIFO is being filled with data from the current frame.
   - The CRC_OK bit no longer reflects the CRC check from the previous frame and now reflects the CRC check for those bytes received in the current frame.
   - The RX_BYTE_COUNT field no longer reflects the count of bytes received in the previous frame, and now reflects the count of bytes received in the current frame.

5.6.1.2 Receive

The receiver continues to accept data and move it to the FIFO as long as RX_DV is active. The receiver does not know about the state of the FIFO; software must monitor the FIFO and remove data to avoid FIFO overflow.

5.6.1.3 Receive End

1. The receiver recognizes deassertion of RX_DV as the end-of-frame delimiter.
2. The RX_EOP_INT interrupt is asserted at the termination of receive data, at which time the receive data is available at the receive FIFO and the count of received bytes is available in RX_BYTE_COUNT.
3. In the case of an odd number of nibbles, the byte count does not include the odd nibble.
4. The CRC is calculated continuously, so the CRC_OK bit reflects the state of the CRC check at the time the signal RX_EOP_INT is asserted.

5.6.1.4 Receive Error – RX FIFO Overflow

During a receive FIFO overflow, the MII receiver continues to receive data and write the data into the receive FIFO without interruption. At the completion of a packet during which there was a receive FIFO overflow, the receive FIFO must be reset (using the RX FIFO Reset bit of the port’s Interrupt Set register).

5.6.1.5 Receive Error – RX_ER asserted

RX_ERR_INT is asserted each time the MII signal RX_ER is asserted. The detection of RX_ER does not affect the reception and movement of data to the receive FIFO. The detection of RX_ER does not cause the receive function to terminate.

5.6.2 Transmit Sequence

The transmitter of the MII controller moves data to the MII interface from the transmit FIFO and signals exception conditions.

5.6.2.1 Transmit Start

1. The first piece of transmit data must be present in the TX FIFO before starting to transmit.
2. The transmit byte count must be set before starting to transmit. The behavior with a transmit byte count of 0 is undefined.
3. The transmit byte count must be present before TX_START is asserted.
4. Transmission is started when the TX_START bit is set.
5. While the transmitter is active, assertion of TX_START produces undefined behavior.

5.6.2.2 Transmit End

1. The transmitter ceases operation after transmitting the number of bytes specified in TX_BYTE_COUNT.
2. The transmitter signals completion by raising TX_EOP_INT.

5.6.2.3 Transmit Error – Collision and Jam

When operating in half duplex mode, if the COL signal is asserted while the MII controller is transmitting, the MII controller:

1. Signals detection of a collision by raising COL_INT.
2. Ceases transmitting the user supplied data and asserts a TX_EOP_INT interrupt.
3. Transmits a jam pattern consisting of the repeating pattern of 1 0 1 0 … (least significant bit first) until a total of 32 bits have been transmitted (8 nibbles of 0x5)
4. Ceases transmission of all data.

The three events, COL_INT, TX_EOP_INT, and jam data transmission, do not occur with deterministic timing, but do obey the following rules:

1. COLLISION and COL_INT are always asserted together and are asserted first.

2. TX_EOP_INT can be asserted as early as one core clock cycle after the assertion of COL_INT and as late as 8 MII tx_clk cycles after the assertion of COL_INT but always synchronous to the core clock.
3. Transmission of the jam sequence is always the last operation to complete.

5.6.2.4 Transmit Error – Premature End of Data

The MII transmitter has no means to validate the contents of the TX FIFO. The transmitter continues to remove data from the FIFO and transmits until it has transmitted TX_BYTE_COUNT bytes. If the FIFO becomes empty during transmission, the transmit FIFO asserts an underflow interrupt.

5.6.3 Physical-Side Mode

Each MII controller can also be configured to operate as the physical side of an MII connection by setting the REVERSE_MII control bit.

In physical-side mode of operation, the physical-side clocks are derived from an internal clock divider. This clock is driven out on the RX_CLK pin for external use.

In physical-side mode, the MII controller:

1. Supplies transmit and receive timing for the interface.
2. Changes the timing reference for the outbound data in order to insure compliance with MII timing.
3. Derives timing from the core clock of the IP3023.
4. Can only operate in full duplex mode, and does not provide any indication of carrier sense or collision.
5. All signals, except for TX_CLK and RX_CLK, operate in the same direction while in MII physical-side mode as when in normal MII mode. Figure 5-16 shows how devices are connected in physical-side mode.

The speed of the interface clock is set by dividing the core clock with a value set through CLK_DIV[7:0]. The interface clock will be set to a speed of clk_core/(CLK_DIV[7:0]+1). If it is desired run the clock at a specific frequency, the core clock must be running at an integer multiple of the desired frequency. The divider used to produce the interface clock from the core clock does not produce a symmetric clock when the division value is an odd number (division value = CLK_DIV[7:0]+1); so, if a symmetric clock is desired, the core clock is further restricted to run at an even integer multiple of the desired target frequency. To set the clock for physical-side MII:

1. Assert reset to the MII controller.
2. Set the desired value for the clock divider.
3. Assert the REVERSE_MII control bit. (Actions 2 and 3 can be reversed)
4. Deassert reset to the MII controller.

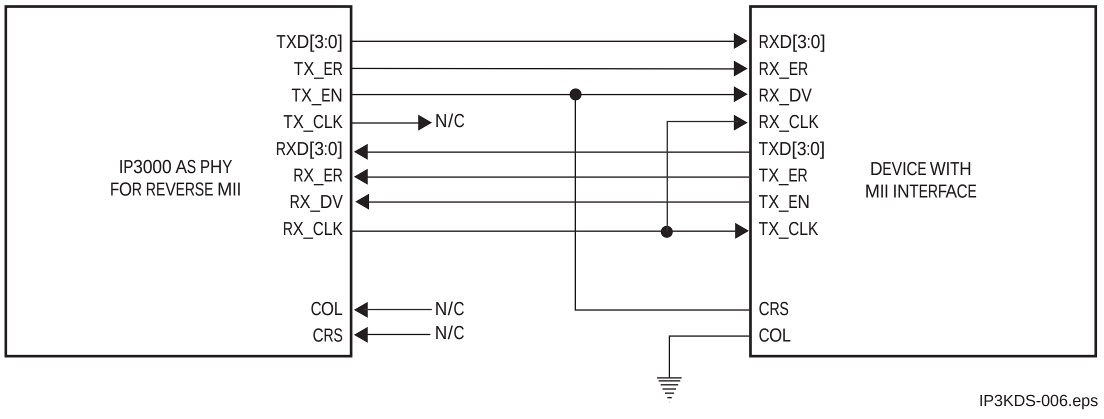


Figure 5-16 Connection Diagram for Physical-Side Mode

5.7 General Purpose Serial Interface (GPSI)

Port E can be used as a high-speed GPSI port. This should not be confused with the SerDes-based GPSI port discussed in Section 5.5.8.

The GPSI seven-signal interface is a de facto standard, not an IEEE standard, so different vendors have varied the implementation slightly. Some variations include the following:

- Polarity of TxEN, RxEN and COL signals
- Relationship of the data (TxD/RxD) to the clock (TxCLK/RxCLK) – which edge is used to drive the data and which edge is used to sample the data
- Setup and hold requirements on the receive portion of the interface
- Propagation delay on the transmit portion of the interface

The implementation in the IP3023 is based on the following assumptions:

- The polarity of TxEN and RxEN is always active high.
- The relationship of TxD to TxCLK and RxD to RxCLK is a subject of uncertainty; therefore it is programmable through programming the clocks' polarities.
- The implementation is not 7-wire, but rather 8-wire. The eighth signal is Carrier Sense (CRS), or Transmit Busy (TxBUSY).
- Since COL and CRS may be active high or low, there is a capability to program how they are sensed.

5.7.1 Interface Signal Description

TxCLK and **RxCLK** – typically a continuous square wave that is supplied by the GPSI master. TxCLK provides the timing reference for the transfer of TxD and TxEN. RxCLK provides the timing reference for the transfer of RxD and RxEN.

TxD – contains the data to be transmitted to the media (by the PHY) and transitions synchronously with respect to TxCLK. It is an input to the PHY from the MAC.

TxEN – indicates to the PHY that the data on TxD should be sampled using TxCLK.

RxD – contains the data received (by the PHY) from the media and transitions synchronously with respect to RxCLK. It is an output from the PHY to the MAC.

RxEN – indicates to the MAC that the data on RxD is valid and can be sampled by RxCLK.

COL and **CRS** (or **TxBUSY**) are used to reflect the status of the transaction. They are generated by the interface master (PHY) and are inputs to the slave (MAC).

Table 5-31 and Table 5-32 show the pin assignments, signal directions, and control functions for this GPSI port. Table 5-31 applies when the IP3023 is the Master, and Table 5-32 applies when the IP3023 is the Slave.

Table 5-31 IP3023 GPSI Master Interface Signal Usage

GPSI Slave Signal Name	IP3023 GPSI Signal Name	Port E Pin	IP3023's Direction	Pin Driver Control
RxEN	TxEN	7	Output	GPIO_EN[7] = 1
RxD	TxD	6	Output	GPIO_EN[6] = 1
RxCLK	TxCLK	5	Output	GPIO_EN[5] = 1
CRS/ TxBUSY	CRS/ TxBUSY	4	Input	GPIO_EN[4] = 0
TxCLK	RxCLK	3	Output	GPIO_EN[3] = 1
TxEN	RxEN	2	Input	GPIO_EN[2] = 0
COL	COL	1	Input	GPIO_EN[1] = 0
TxD	RxD	0	Input	GPIO_EN[0] = 0

Table 5-32 IP3023 GPSI Slave Interface Signal Usage

GPSI Slave Signal Name	IP3023 GPSI Signal Name	Port E Pin	IP3023's Direction	Pin Driver Control
RxEN	TxEN	7	Output	GPIO_EN[7] = 1
RxD	TxD	6	Output	GPIO_EN[6] = 1
RxCLK	TxCLK	5	Input	GPIO_EN[5] = 0
CRS/ TxBUSY	CRS/ TxBUSY	4	Input	GPIO_EN[4] = 0
TxCLK	RxCLK	3	Input	GPIO_EN[3] = 0
TxEN	RxEN	2	Input	GPIO_EN[2] = 0
COL	COL	1	Input	GPIO_EN[1] = 0
TxD	RxD	0	Input	GPIO_EN[0] = 0

5.7.2 Receive Sequence

Receive is the movement of the data into the IP3023, using the RxD, RxEN, and RxCLK pins of the interface.

5.7.2.1 Receive Start

Setting bit RX_ENABLE in the Function Control 1 Register enables the receive of the data.

If the RX_ENABLE bit was set when the RxEN pin was asserted – that is, in the midst of the previous packet – the GPSI receiver will wait until RxEN is deasserted. After that, it becomes ready to receive the next packet.

Prior to setting RX_ENABLE, the following controls must be specified:

- Selection of the master or slave mode and the polarity of the RxCLK (fields MASTER in Function Control 0 and RxCLK_POL in Function Control 1).
- In the case of Master mode, the receive clock divider must be programmed properly (field RxCLK_DIV in Function Control 1).
- SFD_PATTERN and SFD_MASK fields in Function Control 2.

5.7.2.2 Start of Frame Delimiter (SFD)

If SFD_MASK contains all 0s, there will be no Start of Frame Delimiter detection (*no_sfd* mode) and the first data bit accompanied with RxEN active will be considered as valid data. The receive bit counter will start incrementing as soon as RxEN is active as well.

If SFD_MASK is not all 0s (*sfd* mode), a 1 in a mask will allow a corresponding bit from SFD_PATTERN to be used to check against incoming data.

SFD in incoming data is always considered to be 16 bits wide.

If some bits are not needed, they can be masked – that is, if there are only "n" bits that are important, they should be placed in the "n" MSBs of the SFD_PATTERN field, and the "16 - n" LSB bits of the SFD_MASK field may be set to 0.

Example:

If we are only interested in detecting an 8-bit pattern, say AB, then we can program in the following way: SFD_PATTERN = 16'hABxx; SFD_MASK = 16'hFF00, where "x" stands for "any" code.

Still, there MUST be at least 16 bits that precede the first bit of the "body" data.

As soon as the specified pattern is matched, the first bit after that will be considered as valid input data, and the receive bit counter will start incrementing. A one cycle long signal is generated to set the RX_SFD bit in the Interrupt Status register.

5.7.2.3 Receive

The receive process will go on as long as TxEN is asserted.

5.7.2.4 Receive End

When TxEN is deasserted, it is interpreted as the end of the receive.

A one cycle long signal is generated to set Interrupt Status Register bit RX_EOP. The last 32-bit word will be sent to the receive FIFO. This word does not necessary contain 32 meaningful bits. The RX_BIT_CNT field in the Function Status 0 register will reflect the number of the bits received.

Notes:

- The last word will be sent to the RX FIFO not later than when the RX_OEP interrupt signal is issued.
- The code in RX_BIT_CNT[4:0] reflects the number of valid bits within the last 32-bit word, starting from bit 0.

5.7.2.5 Collision

The level on the COL pin is constantly sensed: it is resynchronized to the core clock domain and is represented as a COLLISION bit in Function Status 0. In addition, the transition from inactive to active state (low to high, if COL_POL=0 in Function Control 1, and high to low if COL_POL=1) will generate one cycle long signal that will set the COL bit in the Interrupt Status register.

5.7.2.6 Carrier Sense

The level on the CRS pin is constantly sensed: it is resynchronized to the core clock domain and is represented as a CARRIER_SENSE bit in Function Status 0. In addition, the transition from the active to inactive state (high to low, if CRS_POL=0 in Function Control 1, and low to high if CRS_POL=1) will generate a one cycle long signal that will set the CRS bit in the Interrupt Status register.

5.7.3 Transmit Sequence

Transmit is the movement of the data out of the IP3023, using the TxD, TxEN, and TxCLK pins of the interface.

5.7.3.1 Transmit Start

Prior to the start of transmission, the following must happen:

- At least one entry of the transmit FIFO should be filled with data.
- The transmit bit counter should be set (field TX_BIT_CNT in Function Control 0).
- Selection of the Master or Slave mode and the polarity of the TxCLK must be done (fields MASTER and TxCLK_POL in Function Control 0).
- In the case of Master mode, the transmit clock divider must be programmed properly (field TxCLK_DIV in Function Control 0).

Then, the TX_START bit in the Interrupt Set register should be set. This will generate a one cycle long pulse that will actually start the transmission. The TxEN pin is asserted, together with the first bit of data, and stays asserted until the end of transmission.

Notes:

- The code in the TX_BIT_CNT field represents the number of bits to be transmitted – e.g., 1 means one bit. Therefore, code 0 will cause undefined behavior of the transmit function.
- TX_START should not be issued before the transmission of the previous packet is done, which is indicated by the TX_EOP bit in the Interrupt Status register.

5.7.3.2 Transmit End

When the number of bits specified in the TX_BIT_CNT field have been transmitted, the TxEN pin is deasserted.

Transmission can also be ended earlier by setting the TX_HALT bit in the Interrupt Set register. This will generate a one cycle long pulse that will stop the transmission.

In both cases a one cycle long signal is generated to set bit TX_EOP in the Interrupt Status register.

5.7.4 Master Mode

Master mode is specified by setting the MASTER bit in Function Control 0.

Transmit and receive functionality is the same, except that TxCLK and RxCLK are generated inside the GPSI block, using TxCLK_DIV[7:0] and RxCLK_DIV[7:0].

COL and CRS are not considered as inputs and as such cannot cause an interrupt. Moreover, the IP3023 will not drive any value on these pins.

5.8 External Interrupts

Several pins of the IP3023 are available for general purpose external interrupts. These interrupt pins and their control functions are distributed among the I/O ports as Table 5-33 shows. The interrupt pins and related control register bits of any given port are available regardless of the function selected for that port. These interrupts are level sensitive and must persist for three core clocks.

Table 5-33 General Purpose Interrupt Pins

Port	Port Pin	Interrupt Status Bit	Function Control 2 Configuration Field
B	6	8	9:8
C	13	10	31:30
C	12	9	29:28
D	16	9	31:30
E	3	9	31:30
F	3	9	31:30
G	31	9	31:30
H	15	10	31:30
H	14	9	29:28

For each external interrupt, there is an interrupt configuration field in the Function Control 2 register that controls the behavior of that interrupt. Table 5-34 shows the values of each interrupt configuration register field.

Table 5-34 Interrupt Configuration Field

Value	Function
00	No interrupt.
01	Interrupt on a rising signal.
10	Interrupt on a falling signal.
11	Interrupt on both a rising and a falling signal.

5.9 Timers, Clocks, and Random Number Generator

Timer related functions include:

- Multipurpose Timer
- System Timer
- Random Number Generator
- Core Clock control
- I/O Clock control
- Reset reason flags

Supporting these functions are the following programming interfaces:

- A block of timer registers located in the indirect address space at addresses 0000 0A00–0000 0AFF. Section 6.5 describes those registers.
- INT_STAT0[7:0] – A block of eight interrupts dedicated to the corresponding block of eight 32-bit timer registers. When the value held in a given timer register matches the value of the global cycle count register, the corresponding interrupt is asserted.
- INT_STAT1[27] – Real time compare register interrupt.

5.9.1 Multipurpose Timer / Watchdog

The Multipurpose Timer is a 32-bit free running counter that can be used for recovering from unexpected system hang-ups and for timing real-time events. There are two 32-bit compare registers inside the Multipurpose Timer:

- Watchdog compare register WDCOM. If the Watchdog compare register's output is enabled when its value matches with the Multipurpose Timer's value, the CPU will be reset. The Watchdog compare register can be enabled through the Watchdog Configuration Register WDCFG. If the value 0x4d3c2b1a is written to WDCFG, the output of the Watchdog compare register will be disabled. At reset, the watchdog reset compare register is disabled.
- Real-Time compare register RTCOM. When the Real-Time compare register's value matches with the Multipurpose Timer's value, at the next counter clock the Real-Time Compare Register Interrupt INT_STAT1[27] is asserted (the interrupt mask bit must be set to cause an interrupt).

Since the Multipurpose Timer is typically running much slower than the core, when the Watchdog compare register or the Real-Time compare registers are updated multiple times in a short period, the last value written to those registers will be seen, but some of the intermediate values may be dropped.

The Multipurpose Timer is incremented synchronously with a constant frequency clock that comes from OSC_IN. The time interval during which the Watchdog needs software service in order to not reset the chip can be programmed in the WDCOM register. Once the Watchdog register compare is enabled, software must continually update the WDCOM register, to avoid reset. The counter value of the Multipurpose Timer can be read from the MPTVAL register.

The Multipurpose Timer counter itself is free-running at all times.

The TKEY register provides a protection mechanism to prevent accidental updates to the Watchdog registers WDCOM and WDCFG. If TKEY = 0xa1b2c3d4, then the user can write to WDCOM and WDCFG.

If the Timer key register does not have the correct value, any attempt to write either WDCOM or WDCFG is ignored and does not corrupt the original contents of those registers. Both WDCOM and WDCFG can be read at all times, regardless of TKEY.

5.9.2 System Timer

The System Timer is a 32-bit free running timer intended to generate periodic interrupts for SW drivers that must be called at a constant rate, such as UART and DTMF functions. The counter value of the System Timer can be read from the SYSVAL register.

The System Timer consists of eight 32-bit compare registers (shared among the 8 threads). Each of the compare registers is continually compared against an internal counter. If any of the 8 compare registers' values matches the System Timer's value, at the next counter clock the corresponding System Timer Interrupt (INT_STAT0[7:0]) is asserted (the interrupt mask bit must be set to cause an interrupt).

The System Timer is incremented synchronously with the core clock.

5.9.3 Random Number Generator

The Random Number Generator (RNG) consists of 3 oscillators feeding a Linear Feedback Shift Register (LFSR) after synchronization to the core clock.

Each read of this register returns a uniformly distributed random number. Since the register shifts one bit per core clock, if software needs multiple uncorrelated numbers, it should wait at least 32 core clocks between reads. If software is building a strong cryptographic key longer than 32 bits, it should wait much longer between reads to allow true randomness to accumulate.

The oscillators can be disabled. When their enable bit is high, the oscillator is running, and when it is low it is stopped, with the output in a low state. The oscillator enable bit powers up equal to zero. When the oscillators are disabled, the random number generator still generates uniformly distributed random numbers, but the only source of true randomness is in the timing of the register reads.

5.10 Debug Interface

The IP3023 provides a Debug Interface port to enable communication between an on-chip debug kernel and an external debug monitor. The IP3023's debug port acts as an SPI slave device. Both the debug kernel and the debug monitor are provided by Ubicom as part of the IP3023 development kit. With the debug kernel, the external flash memory can be programmed or updated.

Table 5-35 Debug Interface Signals

Signal Name	I/O	Description
TSCK	I	Serial Clock – This is a clock input for the Debug Interface. It is used to synchronize data movement in and out of the device through TSI and TSO signal lines.
$\overline{\text{TSS}}$	I	Slave Select – This is an active-low select signal which, when asserted, enables the processor to exchange data with the outside devices. This signal must be asserted before data transactions and must stay low for the duration of the transaction. When it is asserted, TSI and TSCK are valid. The transaction is defined as a 40-bit transfer. The $\overline{\text{TSS}}$ signal line must be negated between each 40-bit transfer.
TSI	I	Serial Input – Sampled on the rising edge of TSCK. This is a unidirectional input signal to the processor, and is driven by the SPI master;
TSO	O	Serial Output – Driven after the falling edge of TSCK. This is a unidirectional output signal from the processor. The IP3023 drives this pin only if $\overline{\text{TSS}}$ is held low (TSO is tri-stated otherwise). The TSO pin is driven low if $\overline{\text{TSS}}$ is driven low during reset; TSO will be driven high as soon as the part is out of reset. The MSB bit of the response is an acknowledgement bit; when set to 1, it indicates that the command has been executed.

Internally, the IP3023 moves 40-bit messages between the Debug Interface and the debug mailbox.

The Debug Interface is visible to software as a set of mailbox registers in the global address space. Refer to Table 5-36. The debug kernel retrieves debug commands from the incoming mailbox and deposits results in the outgoing mailbox. The Debug Mailbox Interrupt indicates that a debug message is waiting or has been successfully sent.

Table 5-36 Debug Mailbox Address Map

Address	Register
0000 0B00	Incoming Mailbox
0000 0B04	Outgoing Mailbox – Software should write to the outgoing buffer only when the Empty Flag is asserted.
0000 0B08	Mailbox Status
	31: Incoming Mail Box Full
	30: Incoming Mail Box Empty
	29: Outgoing Mail Box Full
	28: Outgoing Mail Box Empty (write ready)

6.0 Memory Reference

The following sections show addresses, reset values, and descriptions of IP3023 registers.

6.1 Alphabetical List of Registers

Table 6-1 lists the IP3023 registers alphabetically. The subsequent sections describe these registers in functional groups – Per-Thread Registers (Section 6.2), Global Registers (Section 6.3), and Indirect Registers, which are composed of HRT Tables (Section 6.4), Timer Registers (Section 6.5), and Per-Port Registers (Section 6.6).

Table 6-1 Alphabetical List of Registers

Register	Type	Address	Description	Details
A0-A6	Per-Thread	080-098	32-bit address registers	Table 6-2
A7 or SP	Per-Thread	09C	32-bit stack pointer	Table 6-2
BROWN	Indirect	0A2C	Reserved	Table 6-5
CBCFG1	Indirect	0A1C	Core clock configuration	Table 6-5, Figure 3-3
CBCFG2	Indirect	0A20	Serial I/O PLL clock configuration	Table 6-5, Figure 3-3
CHIP_ID	Global	100	Chip Device ID and Revision Number	Table 6-3, Section 6.3.1
CSR	Per-Thread	0B4	Control and Status Register	Table 6-2, Section 6.2.1
D0-D15	Per-Thread	000-03C	General-purpose data registers	Table 6-2
DCAPT	Global	170	Data capture address	Table 6-3, Section 6.3.5
DCAPT_PC	Global	174	Program counter value when DCAPT interrupt set	Table 6-3
DCAPT_TNUM	Global	178	Thread ID and cause when DCAPT interrupt set	Table 6-3, Section 6.3.6
Function	Indirect, Per-Port	00 + offset	Function select, reset, and FIFO configuration	Table 6-7, Figure 6.6.1
Function Ctl 0	Indirect, Per-Port	28 + offset	See Section 6.7 through Section 6.14.	Table 6-7
Function Ctl 1	Indirect, Per-Port	2C + offset	See Section 6.7 through Section 6.14.	Table 6-7
Function Ctl 2	Indirect, Per-Port	30 + offset	See Section 6.7 through Section 6.14.	Table 6-7
Function Status 0	Indirect, Per-Port	34 + offset	See Section 6.7 through Section 6.14.	Table 6-7
Function Status 1	Indirect, Per-Port	38 + offset	See Section 6.7 through Section 6.14.	Table 6-7
GLOBAL_CTRL	Global	134	Global Control Register	Table 6-3
GPIO Ctl	Indirect, Per-Port	04 + offset	GPIO output enable	Table 6-7
GPIO In	Indirect, Per-Port	0C + offset	GPIO In – The current state of the external I/O pins	Table 6-7

Table 6-1 Alphabetical List of Registers

Register	Type	Address	Description	Details
GPIO Out	Indirect, Per-Port	08 + offset	GPIO data out	Table 6-7
HRT0	Indirect, HRT	800-83C	Hard Real-Time Table 0 (Sixteen 32 bit registers)	Table 6-4
HRT1	Indirect, HRT	900-93C	Hard Real-Time Table 1 (Sixteen 32 bit registers)	Table 6-4
INST_CNT	Per-Thread	0B0	Executed instruction count	Table 6-2
INT_CLR0	Global	124	Clears bits in INT_STAT0.	Table 6-3
INT_CLR1	Global	128	Clears bits in INT_STAT1.	Table 6-3
INT_MASK0	Per-Thread	0C0	AND’ed with Global INT_STAT0 to mask interrupts	Table 6-2, Section 6.2.3
INT_MASK1	Per-Thread	0C4	AND’ed with Global INT_STAT1 to mask interrupts	Table 6-2, Section 6.2.4
INT_SET0	Global	114	Sets bits in INT_STAT0.	Table 6-3
INT_SET1	Global	118	Sets bits in INT_STAT1.	Table 6-3
INT_STAT0	Global	104	Interrupt Status Register 0	Table 6-3, Section 6.3.2.
INT_STAT1	Global	108	Interrupt Status Register 1	Table 6-3, Section 6.3.3.
Interrupt Clear	Indirect, Per-Port	1C + offset	Port interrupt clear	Table 6-7, Section 6.6.2
Interrupt Mask	Indirect, Per-Port	14 + offset	Port interrupt mask	Table 6-7, Section 6.6.2
Interrupt Set	Indirect, Per-Port	18 + offset	Port interrupt set	Table 6-7, Section 6.6.2
Interrupt Status	Indirect, Per-Port	10 + offset	Port interrupt status	Table 6-7, Section 6.6.2
IREAD_DATA	Per-Thread	0BC	IREAD instruction output	Table 6-2
MAC_HI	Per-Thread	0A0	High 32 bits of multiplier accumulate/result	Table 6-2
MAC_LO	Per-Thread	0A4	Low 32 bits of multiplier accumulate/result	Table 6-2
MAC_RC16	Per-Thread	0A8	Rounded and clipped multiplier accumulate/result	Table 6-2
MPIMCTRL	Indirect	0A30	Mpim BIST control register’s value	Table 6-5
MPIMSTAT	Indirect	0A34	Mpim BIST statics register’s value	Table 6-5
MPTVAL	Indirect	0A00	Multipurpose Timer value	Table 6-5
MT_ACTIVE	Global	138	Active/inactive status for each thread	Table 6-3
MT_ACTIVE_CLR	Global	140	Clears bits in MT_ACTIVE.	Table 6-3
MT_ACTIVE_SET	Global	13C	Sets bits in MT_ACTIVE.	Table 6-3
MT_BREAK	Global	158	Multithreading BKPT Executed	Table 6-3
MT_BREAK_CLR	Global	15C	Clears bits in MT_BREAK.	Table 6-3
MT_DBG_ACTIVE	Global	144	AND’ed with MT_ACTIVE for debug control.	Table 6-3
MT_DBG_ACTIVE_CLR	Global	17C	Clears bits in MT_DBG_ACTIVE.	Table 6-3
MT_DBG_ACTIVE_SET	Global	148	Sets bits in MT_DBG_ACTIVE.	Table 6-3

Table 6-1 Alphabetical List of Registers

Register	Type	Address	Description	Details
MT_EN	Global	14C	Multithreading Enable	Table 6-3
MT_HPRI	Global	150	Multithreading High-Priority Thread	Table 6-3
MT_HRT	Global	154	Multithreading Hard Real-Time Thread	Table 6-3
MT_MIN_DELAY_EN	Global	164	Multithreading Minimum Delay Enable	Table 6-3
MT_SINGLE_STEP	Global	160	Controls single-step operation for each thread.	Table 6-3
PC	Per-Thread	0D0	32-bit program counter	Table 6-2
PERR_ADDR	Global	16C	Address of a reported memory parity error	Table 6-3
ROSR	Per-Thread	0B8	Read-only Status Register	Table 6-2, Section 6.2.2
RSGCFG	Indirect	0A04	Random Number Generator configuration	Table 6-5
RSTFLAG	Indirect	0A24	Reset reason flags loaded during reset	Table 6-5
RTCOM	Indirect	0A08	Real-Time Compare register	Table 6-5
RX FIFO	Indirect, Per-Port	24 + offset	See Section 6.7 through Section 6.14.	Table 6-7
SCRATCHPAD0	Global	180	Scratch Pad Register 0	Table 6-3
SCRATCHPAD1	Global	184	Scratch Pad Register 1	Table 6-3
SCRATCHPAD2	Global	188	Scratch Pad Register 2	Table 6-3
SCRATCHPAD3	Global	18C	Scratch Pad Register 3	Table 6-3
SOURCE3	Per-Thread	0AC	Implicit third source operand	Table 6-2
SP or A7	Per-Thread	09C	32-bit stack pointer	Table 6-2
SYSCOM0	Indirect	0A80	System Timer Compare Register 0 (INT_STAT0[0])	Table 6-5
SYSCOM1	Indirect	0A84	System Timer Compare Register 0 (INT_STAT0[1])	Table 6-5
SYSCOM2	Indirect	0A88	System Timer Compare Register 0 (INT_STAT0[2])	Table 6-5
SYSCOM3	Indirect	0A8C	System Timer Compare Register 0 (INT_STAT0[3])	Table 6-5
SYSCOM4	Indirect	0A90	System Timer Compare Register 0 (INT_STAT0[4])	Table 6-5
SYSCOM5	Indirect	0A94	System Timer Compare Register 0 (INT_STAT0[5])	Table 6-5
SYSCOM6	Indirect	0A98	System Timer Compare Register 0 (INT_STAT0[6])	Table 6-5
SYSCOM7	Indirect	0A9C	System Timer Compare Register 0 (INT_STAT0[7])	Table 6-5
SYSVAL	Indirect	0A18	The value of the System Timer	Table 6-5
TKEY	Indirect	0A0C	Timer Block’s security key code	Table 6-5
TRN	Indirect	0A28	32-bit Random Number	Table 6-5
TX FIFO	Indirect, Per Port	20 + offset	See Section 6.7 through Section 6.14.	Table 6-7
WDCFG	Indirect	0A14	Watchdog Configuration register	Table 6-5
WDCOM	Indirect	0A10	Watchdog Compare register	Table 6-5

6.2 Per-Thread Registers

Table 6-2 Per-Thread Registers

Address	Register	Read/ Write	Description	32-Bit Reset Value
000-03C	D0–D15	R/W	General-purpose 32-bit data registers. The only restriction is that they cannot be used as address or stack pointer registers.	0000 0000
040-07C	Reserved			
080-098	A0–A6	R/W	32-bit address registers. These are used as pointers to operands.	0000 0000
09C	A7 or SP	R/W	32-bit Stack pointer, also referred to as A7. The use of A7 as the stack pointer register is conventional, not “hard-wired” in the architecture. There are no instructions that use SP explicitly.	0000 0000
0A0	MAC_HI	R/W	High 32 bits of multiply-accumulate (MAC) and multiplier result. Also set by CRCGEN.	0000 0000
0A4	MAC_LO	R/W	Low 32-bits of MAC and multiplier result. Also set by CRCGEN.	0000 0000
0A8	MAC_RC16	R/W	Rounded and Clipped S16.15 format of the most recent MAC, multiplier, CRCGEN result. The 48 bit result is interpreted as s16.31 format and rounded/clipped to S.15 format. This is then sign extended to 32 bits.	0000 0000
0AC	SOURCE3	R/W	Used as an implicit third source operand by certain instructions. Since the instruction formats are limited to 2 source and one destination operand, this register is used when a third operand is needed.	0000 0000
0B0	INST_CNT	RO	This register maintains a count of the executed instructions for the associated context. When the execution of a context is suspended, the associated counter also stops. This 32-bit count starts as zero after a reset operation, and otherwise cannot be reset. The count rolls over when it reaches its maximum value of 0xFFFF FFFF. This count, in conjunction with the global timer value, can be used to determine run-time performance distribution statistics, as well as to help in debugging (e.g., did thread #N execute any instructions yet?). Reading INST_CNT for a thread that is not quiescent gives an approximate value.	0000 0000
0B4	CSR (Control & Status Register)	R/W	Contains condition codes, and other status bits, as well as some thread-specific control bits (refer to Section 6.2.1)	0000 0000
0B8	ROSR (Read-Only Status Register)	RO	Extension of CSR containing bits or fields that are set by hardware, but read-only by software (refer to Section 6.2.2).	0000 0000
0BC	IREAD_DATA	R/W	The IREAD instruction output is placed in this 32-bit register.	0000 0000

Table 6-2 Per-Thread Registers

Address	Register	Read/ Write	Description	32-Bit Reset Value
0C0	INT_MASK0	R/W	A 32-bit mask that, when AND'ed with the global 32-bit INT_STAT0 register, determines whether an interrupt condition is seen by a given thread (refer to Section 6.2.3).	0000 0000
0C4	INT_MASK1	R/W	A 32-bit mask that, when AND'ed with the global 32-bit INT_STAT1 register, determines whether an interrupt condition is seen by a given thread (refer to Section 6.2.4).	0000 0000
0C8-0CC	Reserved			
0D0	PC	R/W	32-bit Program Counter. Only valid for access in a thread that is quiescent (has no instructions in the pipeline). In that case, it points to the next instruction to be executed when that thread resumes. The PC is set by one of the following:· • Set to the power-up address after reset/power-up. • Direct write by another thread, when this thread is inactive. The controlling thread uses the destination thread select field in its CSR to address the target thread's PC register. Do not try to set PC of the current thread to do a jump.	0000 0000
0D4-0FC	Reserved			

For those per-thread registers that have special functions assigned to bits or fields within the register, those bits and fields are described below.

6.2.1 CSR

The Control and Status Register contains condition codes and other status bits, as well as some thread-specific control bits as follows:

Bits	Description	
31:20	Reserved	
19:14	Destination Thread Select. Used to override the default context selection for directly addressed destination operand register. Has no effect for address calculations. Does not affect implicit destinations (such as MAC_HI). Bits are defined as follows:	
	19:15	Destination context number. Relevant only if bit 14 is set.
	14	Enable: 1 On; 0 Off.
13:8	Source Thread Select. Used to override the default context selection for directly addressed source 1 operand register. Has no effect for address calculations. Bits are defined as follows:	
	13:9	Source context number. Relevant if bit 8 is set.
	8	Enable source thread override: 1 On; 0 Off.
7-4	32-bit-Operand Condition Code Bits as {N,Z,V,C}32 ordering. Valid only when the thread has no instructions in the pipeline.	
3-0	16-bit-Operand Condition Code Bits as {N,Z,V,C}16 ordering. Valid only when the thread has no instructions in the pipeline.	

Note: Writing the source or destination thread select fields will also modify NZVC bits of the current thread at an unpredictable time.

6.2.2 ROSR

The Read-Only Status Register is an extension of CSR that contains bits or fields that are set by hardware, but read-only by software.

Bits	Description
31:7	Reserved
6:5	Reserved
4:2	THREAD NUMBER: Identifies the current context.
1	MEM BUSY: When set to '1', it indicates that this thread has an uncompleted IREAD or IWRITE operation in progress. If the uncompleted operation is an IREAD, indicates that IREAD_DATA register is invalid. If an IREAD or IWRITE instruction is executed when the Mem Busy bit it set, the result is undefined.
0	INTERRUPT CONDITION. Indicates a pending interrupt condition for the thread. Derived by AND'ing the 64-bit thread-specific mask with the general interrupt status bits.

6.2.3 INT_MASK0

This 32-bit register defines a 32-bit mask, which when ANDed with the global 32-bit INT_STAT0 register, determines whether an interrupt condition is seen by a given thread.

Bits	Description
31:8	Mask for 24 software interrupts
7:0	Mask for 8 System Timer interrupts

6.2.4 INT_MASK1

This 32-bit register defines a 32-bit mask, which when ANDed with the global 32-bit INT_STAT1 register determines whether an interrupt condition is seen by a given thread.

Bits	Description
31	Mask for Breakpoint interrupt
30	Mask for Debug Port Mailbox interrupt
29	DCAPT interrupt mask
28	Reserved
27	Mask for Real-Time compare register interrupt
26	Mask for memory parity error interrupt
25:24	Reserved
23:0	Masks for I/O interrupts

6.3 Global Registers

All these registers are 32 bits wide.

Table 6-3 Global Registers

Address	Register(s)	Read/ Write	Description	32-Bit Reset Value
100	CHIP_ID	RO	Chip Device ID and Revision Number (refer to Section 6.3.1).	0001 0000
104 108	INT_STAT0 INT_STAT1	RO	These two 32-bit registers contain 64-bits of hardware and software generated interrupt conditions. A hardware interrupt condition could, for example, be triggered by a timer condition whose interrupts are enabled (refer to Section 6.3.2 and Section 6.3.3).	0000 0000
10C–110	Reserved			

Table 6-3 Global Registers

Address	Register(s)	Read/ Write	Description	32-Bit Reset Value
114 118	INT_SET0 INT_SET1	WO	When a value is written to one of these 32-bit registers, each bit position containing a 1 causes the corresponding bit in the INT_STAT0 or INT_STAT1 register to be set, unless it is an I/O interrupt bit.	0000 0000
11C–120	Reserved			
124 128	INT_CLR0 INT_CLR1	WO	When a value is written to one of these 32-bit registers, each bit position containing a 1 causes the corresponding bit in the INT_STAT0 or INT_STAT1 register to be cleared, unless it is an I/O interrupt bit.	0000 0000
12C–130	Reserved			
134	GLOBAL_CTRL	R/W	This register contains miscellaneous control bits and values. The control bits act as global enable control for certain functions of the processor. Setting one of these enable bits to one acts as an overall enable for the function, but other registers are still involved in the detailed operation of the functions.	0000 0000
138	MT_ACTIVE	RO	This 32-bit register controls the thread's active/inactive status. The register has one status bit per thread; bit position equals thread number (bits 31:8 are reserved). Bits are cleared by a suspend instruction or write to MT_ACTIVE_CLR; bits are set by an unmasked interrupt or write to MT_ACTIVE_SET. 0 = Suspended 1 = Active	0000 0000
13C	MT_ACTIVE_SET	WO	When a value is written to this register, each bit position containing a 1 causes the corresponding bit in the MT_ACTIVE register to be set (bits 31:8 are reserved).	0000 0000
140	MT_ACTIVE_CLR	WO	When a value is written to this register, each bit position containing a 1 causes the corresponding bit in the MT_ACTIVE register to be cleared (bits 31:8 are reserved).	0000 0000
144	MT_DBG_ACTIVE	RO	This register is AND'ed with MT_ACTIVE register, for purposes of thread scheduling. It prevents threads that are halted for debugging from being inadvertently reactivated by the occurrence of an interrupt. Bits in this register are cleared by break conditions or by single step operations or writing to MT_DBG_ACTIVE_CLR. Once cleared, they can only be set by software, by writing to the MT_DBG_ACTIVE_SET register. (Bits 31:8 are reserved.)	0000 0001
148	MT_DBG_ACTIVE_SET	WO	Writing a 1 into a given bit position causes the corresponding bit in the MT_DBG_ACTIVE register to be set (bits 31:8 are reserved).	0000 0000

Table 6-3 Global Registers

Address	Register(s)	Read/ Write	Description	32-Bit Reset Value
14C	MT_EN (Multithreading Enable)	R/W	Indicates which threads are currently enabled. If a thread is not enabled, then it cannot be allocated any execution slots. If it is enabled, it could either be suspended (inactive) or running. Clearing a thread's enable bit blocks further execution of the thread, but does not clear its bit in the MT_ACTIVE register. Bit position corresponds to thread number (bits 31:8 are reserved). 0 = Disabled 1 = Enabled	0000 0001 (Thread #0 enabled, all other threads disabled)
150	MT_HPRI (Multithreading High Priority Thread)	R/W	32-bit register that determines the priority of Non-Real-Time (NRT) threads as high or low priority. For HRT threads this bit is ignored. Bit position corresponds to thread number (bits 31:8 are reserved). 0 = Low 1 = High	0000 0001
154	MT_HRT (Multithreading Hard Real-Time Thread)	R/W	32-bit register that determines which threads are HRT and NRT: Bit position corresponds to thread number (bits 31:8 are reserved). 0 = NRT 1 = HRT	0000 0000
158	MT_BREAK (Multithreading BKPT executed)	RO	32-bit register that indicates which thread or threads are halted for a break condition of some sort (1 = halted). The BKPT instruction can cause an arbitrary number of threads to be halted, but only the bit for the thread executing the instruction is set in this register. It can therefore be read to determine which thread executed the BKPT instruction (bits 31:8 are reserved).	0000 0000
15C	MT_BREAK_CLR	WO	Writing a 1 to any bit in this register causes the corresponding bit in the multithreading break register to be cleared (bits 31:8 are reserved).	0000 0000
160	MT_SINGLE_STEP	R/W	A 1 bit in this register causes the corresponding thread to be allocated exactly one pipeline slot the next time it is scheduled. After that, the scheduling hardware immediately clears its enable bit in the MT_DBG_ACTIVE register. This blocks further scheduling of the thread until its bit in the MT_DBG_ACTIVE register is set by writing a 1 to its position in the MT_DBG_ACTIVE_SET register (bits 31:8 are reserved).	0000 0000

Table 6-3 Global Registers

Address	Register(s)	Read/ Write	Description	32-Bit Reset Value
164	MT_MIN_DELAY_EN (MT Minimum Delay Enable)	R/W	Specifies threads whose scheduling is constrained by the minimum delay interval specified in the GLOBAL_CTRL register. The minimum delay feature can be used for silicon debugging, if a problem related to hazard detection and forwarding paths is suspected. It can also be used to provide more deterministic behavior to threads whose coding leaves them subject to pipeline hazards (bits 31:8 are reserved).	0000 0000
168	Reserved			
16C	PERR_ADDR	RO	Address of a reported memory parity error	xxxx xxxx
170	DCAPT	R/W	Data Capture Address (refer to Section 6.3.5).	0000 0000
174	DCAPT_PC	RO	The Program Counter value captured when the DCAPT interrupt bit was set.	0000 0000
178	DCAPT_TNUM	RO	Thread ID and cause captured when the DCAPT interrupt bit was set (refer to Section 6.3.6).	0000 0000
17C	MT_DBG_ACTIVE_CLR	WO	Writing a 1 into a given bit position causes the corresponding bit in the MT_DBG_ACTIVE register to be cleared.	0000 0000
180	SCRATCHPAD0	R/W	Four scratch-pad registers. Cleared at power-on and by any other reset.	0000 0000
184	SCRATCHPAD1	R/W		
188	SCRATCHPAD2	R/W		
18C	SCRATCHPAD3	R/W		
190–3FC	Reserved			

For those global registers that have special functions assigned to bits or fields within the register, the definitions of those bits and fields are described below.

6.3.1 CHIP_ID

Bits	Description	Read-Only Value
31:16	Chip ID	0001
15:0	Revision number	0000

6.3.2 INT_STAT0

Bits	Description
31:8	24 software interrupts
7:0	8 System Timer interrupts

6.3.3 INT_STAT1

Bits	Description
31	Breakpoint interrupt
30	Debug Port Mailbox interrupt
29	DCAPT interrupt
28	Coprocessor interrupt
27	Real-Time compare register interrupt
26	Memory parity error interrupt
25:24	Reserved
23	I/O Port H, Interrupt 2
22	I/O Port H, Interrupt 1
21	I/O Port H, Interrupt 0
20	I/O Port G, Interrupt 2
19	I/O Port G, Interrupt 1

Bits	Description
18	I/O Port G, Interrupt 0
17	I/O Port F, Interrupt 2
16	I/O Port F, Interrupt 1
15	I/O Port F, Interrupt 0
14	I/O Port E, Interrupt 2
13	I/O Port E, Interrupt 1
12	I/O Port E, Interrupt 0
11	I/O Port D, Interrupt 2
10	I/O Port D, Interrupt 1
9	I/O Port D, Interrupt 0
8	I/O Port C, Interrupt 2
7	I/O Port C, Interrupt 1
6	I/O Port C, Interrupt 0
5	I/O Port B, Interrupt 2
4	I/O Port B, Interrupt 1
3	I/O Port B, Interrupt 0
2	I/O Port A, Interrupt 2
1	I/O Port A, Interrupt 1
0	I/O Port A, Interrupt 0

I/O interrupt bits in INT_STAT1 23:0 give the status of the associated interrupt bits in I/O blocks and can not be set or cleared directly using INT_SET1 or INT_CLR1. These bits correspond to port interrupt conditions as follows:

- Interrupt 0 – Receive FIFO high watermark condition
- Interrupt 1 – Transmit FIFO low watermark condition
- Interrupt 2 – All other port interrupt conditions

6.3.4 GLOBAL_CTRL

Bits	Description
31:11	Reserved
10	PGEN_FORCE – Parity force value
9	PGEN_EN – Parity generation enable
8	PERR_EN – Parity error reset enable
7	Reserved
6:3	Minimum Instruction Delay; minimum number of cycles between successive execution slots for a thread. This setting applies to threads whose (MT_MIN_DELAY_EN bit is set.
2	HRT Table Select 0 = Table 0 1 = Table 1
1	Reserved
0	INT_EN – Interrupt Enable

6.3.5 DCAPT (Data Capture Address)

A write operation (including IWRITE) to the memory address or direct register specified by the DCAPT register will trigger capture of the PC and thread ID of the instruction responsible and set a bit in INT_STAT1.

Bits	Description		
0	= 0	Match indirect, memory address	
		Bits	Description
		31:2	Memory word address
0	= 1	Match direct register address	
		Bits	Description
		31:13	Must be zero
		12:10	Thread ID (0 for global registers)
		9:2	Register number (eight most significant bits of the register address)

This register is always enabled. However, it can be loaded with a value that never matches. A value that can never match is one that has bit 0 equal to 1 and bits 31:13 not equal to all 0's; for example: 0x8000 0001.

6.3.6 DCAPT_TNUM

Thread ID and cause captured when the DCAPT interrupt bit was set.

Bits	Description
9:5	Cause of event 9: Unaligned destination 8: Unaligned source 7: Destination illegal address 6: Source illegal address 5: DCAPT target address match
4:3	Reserved
2:0	Thread ID

6.4 HRT Tables

There are two HRT tables:

- HRT0 at addresses 00000800–0000083c
- HRT1 at addresses 00000900–0000093c

One of the two HRT tables is active and being used by the CPU; the other is available for updates. The HRT Table Select bit in the GLOBAL_CTRL register determines which is the active table.

Each HRT table is composed of 64 8-bit entries. Four 8-bit entries are packed into a 32-bit word, as follows:

Table 6-4 HRT Word (Four HRT Table Entries)

Name	Bits	Read/ Write	Description	Reset value
HRT_END0	31	RW	1 indicates end of HRT table	c0c0c0c0
HRT_NO_THREAD0	30	RW	1 indicates unoccupied entry	
			0 indicates occupied entry	
	29:27	Reserved		
HRT_TNUM0	26:24	RW	HRT thread number	
HRT_END1	23	RW	1 indicates end of HRT table	
HRT_NO_THREAD1	22	RW	1 indicates unoccupied entry	
			0 indicates occupied entry	
	21:19	Reserved		
HRT_TNUM1	18:16	RW	HRT thread number	
HRT_END2	15	RW	1 indicates end of HRT table	
HRT_NO_THREAD2	14	RW	1 indicates unoccupied entry	
			0 indicates occupied entry	
	13:11	Reserved		
HRT_TNUM2	10:8	RW	HRT thread number	
HRT_END3	7	RW	1 indicates end of HRT table	
HRT_NO_THREAD3	6	RW	1 indicates unoccupied entry	
			0 indicates occupied entry	
	5:3	Reserved		
HRT_TNUM3	2:0	RW	HRT thread number	

6.5 Timers, Clocks, and RNG Registers

A block of timer registers located in the indirect address space at addresses 0000 0A00–0000 0AFF control a variety of functions, including:

- Multipurpose Timer
- System Timer
- Random Number Generator (RNG)
- Core Clock control
- I/O Clock control
- Reset reason flags

Table 6-5 Timer Register Definition

Address Offset (+0A00)	Register Name	Bits	Description	Read/ Write	32-Bit Reset Value
00	MPTVAL	31:0	The value of the Multipurpose Timer	RO	0000 0000
04	RSGCFG	31:6	Reserved	R/W	0000 0000
		5:3	Reserved		
		2	Enable Random Number Generator Slow Oscillator 0 = Disable 1 = Enable		
		1	Enable Random Number Generator Medium Oscillator 0 = Disable 1 = Enable		
		0	Enable Random Number Generator Fast Oscillator 0 = Disable 1 = Enable		
08	RTCOM	31:0	Real-Time Compare Register	R/W	0000 0000
0C	TKEY	31:0	Timer Block's security key code. If TKEY = 0xa1b2c3d4, then the user can write to WDCOM and WDCFG.	R/W	0000 0000
10	WDCOM	31:0	Watchdog Compare Register. This register can be written only if TKEY has the correct value.	R/W	0000 0000
14	WDCFG	31:0	Watchdog Configuration Register. This register can be written only if TKEY has the correct value. 0x4d3c2b1a = Disable the Watchdog Register compare. Other value = Enable the Watchdog Register compare.	R/W	4d3c 2b1a
18	SYSVAL	31:0	The value of the System Timer	RO	0000 0000

Table 6-5 Timer Register Definition

Address Offset (+0A00)	Register Name	Bits	Description	Read/ Write	32-Bit Reset Value
1C	CBCFG1	Core Clock Configuration (see also Figure 3-3)		R/W	0000 0080
		31	Reserved.		0
		30:29	Reserved		0
		28:23	Core clock PLL ref divider		0
		22:11	Core clock PLL feedback divider		0
		10:8	Core clock PLL output divider		0
		7	Core clock reset - This bit must be set while changing any bits 28:8, and stay set for at least 5 µs after any bits 28:8 are changed. 0 = normal operation 1 = reset - the PLL will output between 5MHz and 200MHz into the output divider, unless bit 5 is set.		1
		6	PLL bypass select 0 = Core PLL selected 1 = OSC_IN pin divided by 2 selected		0
		5	Core clock PLL power down 0 = normal operation 1 = All analog circuitry in the PLL is turned off, so as to dissipate only leakage current. PLL stabilization time is 1000 cycles - must wait at least 1000 cycles into the PLL (out of the Ref. clock divider) after changing this bit 5 from 1 to 0, before changing bit 4 from 0 to 1.		0
		4	Core clock source select - this bit must = 0 while changing any of bits 28 through 5. 0 = OSC_IN pin 1 = PLL		0
		3:0	Forward Divider. Divisor value used to generate CLK_CORE from the PLL. The PLL output is divided by the value of Forward Divider + 1, allowing for divide values from 1 through 16.		0

Table 6-5 Timer Register Definition

Address Offset (+0A00)	Register Name	Bits	Description	Read/ Write	32-Bit Reset Value
20	CBCFG2	Serial I/O PLL Clock configuration (see also Figure 3-3)		R/W	0000 0080
		31	Serial I/O clock ref select 0 = reference clock = OSC_IN pin 1 = auxiliary serial I/O clock input (PD17)		0
		30:29	Reserved		0
		28:23	Serial I/O clock PLL ref divider		0
		22:11	Serial I/O clock PLL feedback divider		0
		10:8	Serial I/O clock PLL output divider		0
		7	Serial I/O clock reset - This bit must be set while changing any bits 28:8, and stay set for at least 5 µs after any bits 28:8 are changed. 0 = normal operation 1 = reset - the PLL will output between 5MHz and 200MHz into the output divider, unless bit 5 is set.		1
		6	Serial I/O PLL bypass select 0 = PLL selected 1 = OSCSD/2 selected		0
		5	Serial I/O PLL power down 0 = normal operation 1 = All analog circuitry in the PLL is turned off, so as to dissipate only leakage current. PLL stabilization time is 1000 cycles - must wait at least 1000 cycles into the PLL (out of the Ref. clock divider) after changing this bit 5 from 1 to 0, before changing bit 4 from 0 to 1.		0
		4	Serial I/O clock source select - this bit must = 0 while changing any of bits 28 through 5. 0 = OSCSD 1 = SerDes PLL		0
		3:0	Reserved		0

Table 6-5 Timer Register Definition

Address Offset (+0A00)	Register Name	Bits	Description	Read/ Write	32-Bit Reset Value
24	RSTFLAG		Reset reason flags loaded during reset.	RO	0000 0uuu u=undefined
		31:11	Reserved		
		10	Package type 0 = 228 Pin BGA		
		9	Reserved		
		8	Watchdog reset reason flag		
		7	Parity error reset reason flag		
		6	Debug reset reason flag		
		5	Power on reset reason flag - If the $\overline{\text{RST}}$ pin is held low during power up, both the power-on reset reason flag and the $\overline{\text{RST}}$ pin reason flag will be set.		
		4	$\overline{\text{RST}}$ pin reset reason flag		
		3:0	Reserved		
28	TRN	31:0	32-bit Random Number	RO	FFFF FFFF
2C	BROWN	31:3	Reserved	RO	FFFF FFFF
		2:0	Do not change from reset value of 111.	R/W	
30	MPIMCTRL	31:0	Instuction Memory BIST control register's value. The user can write to this register only if TKEY has the correct value.	R/W	0000 0000
34	MPIMSTAT	31:0	Instruction Memory BIST status register's value.	RO	0000 0000
38–7C			Reserved		0000 0000
80	SYSCOM0	31:0	System Timer Compare Register 0 (INT_STAT0[0])	R/W	0000 0000
84	SYSCOM1	31:0	System Timer Compare Register 1 (INT_STAT0[1])	R/W	0000 0000
88	SYSCOM2	31:0	System Timer Compare Register 2 (INT_STAT0[2])	R/W	0000 0000
8c	SYSCOM3	31:0	System Timer Compare Register 3 (INT_STAT0[3])	R/W	0000 0000
90	SYSCOM4	31:0	System Timer Compare Register 4 (INT_STAT0[4])	R/W	0000 0000
94	SYSCOM5	31:0	System Timer Compare Register 5 (INT_STAT0[5])	R/W	0000 0000
98	SYSCOM6	31:0	System Timer Compare Register 6 (INT_STAT0[6])	R/W	0000 0000
9C	SYSCOM7	31:0	System Timer Compare Register 7 (INT_STAT0[7])	R/W	0000 0000
A0–FC			Reserved		0000 0000

6.6 Per-Port Registers

Each port has a set of registers that constitute the primary input, output, and control interfaces for the port. The registers of all ports are similar in number, general format, and general usage, but differ in many respects from port to port and from function to function within each port. Table 6-7 outlines the set of registers available for each port. The address of each register is the sum of the port base address (refer to Table 6-6) and the register address offset (Table 6-7).

Some functions are common to all or most of the ports; and those functions are controlled by corresponding registers and register fields in all the ports. The common functions are primarily those dealing with function selection and FIFO management.

Table 6-6 I/O Port Base Addresses

Port	Address Range
A	0000 1000 – 0000 107F
B	0000 1080 – 0000 10FF
C	0000 1100 – 0000 117F
D	0000 1180 – 0000 11FF
E	0000 1200 – 0000 127F
F	0000 1280 – 0000 12FF
G	0000 1300 – 0000 137F
H	0000 1380 – 0000 13FF

Table 6-7 Per-Port Registers

Offset	Register	Read/ Write	Description	32-Bit Reset Value
0x00	Function	R/W	Function select, reset, and FIFO configuration. Refer to Section 6.6.1	0000 0000
0x04	GPIO Ctl	R/W	GPIO Output Enable. A 1 in any bit position enables the corresponding output buffer.	0000 0000
0x08	GPIO Out	R/W	GPIO Data Out. With the corresponding output pin enabled, the value in this register is driven by the enabled output buffer.	0000 0000
0x0c	GPIO In	RO	GPIO In. This register reflects the current state of the external I/O pins. The function of this register is unaffected by any register settings.	xxxx xxxx
0x10	Interrupt Status	RO	Refer to Section 6.6.2.	xxxx xxxx
0x14	Interrupt Mask	R/W	Refer to Section 6.6.2.	0000 0000
0x18	Interrupt Set	WO	Interrupt set & one cycle pulse generation. Refer to Section 6.6.2.	0000 0000
0x1c	Interrupt Clear	WO	Refer to Section 6.6.2.	0000 0000
0x20	TX FIFO	WO	Refer to individual port and function definitions in sections 6.7 through 6.14.	N/A
0x24	RX FIFO	RO		xxxx xxxx
0x28	Function Ctl 0	R/W		0000 0000
0x2c	Function Ctl 1	R/W		0000 0000
0x30	Function Ctl 2	R/W		0000 0000
0x34	Function Status 0	RO		xxxx xxxx
0x38	Function Status 1	RO	Refer to Section 6.6.3.	xxxx xxxx

6.6.1 Port Function Select Register

Bits	Field Name	Description
31:30	Reserved	
29:24	TX Set	Transmit FIFO watermark trigger level
23:22	Reserved	
21:16	RX Set	Receive FIFO watermark trigger level
15:8	Reserved	
7:4	Function Reset	Asserting these bits resets the corresponding functions. These bits are static and must be deasserted in order to allow the function to come out of reset. Bit 7: Reset function 3 Bit 6: Reset function 2 Bit 5: Reset function 1 Bit 4: Reset function 0
3	Receive FIFO Select	Select the receive FIFO to access. 0 = Receive FIFO 0 1 = Receive FIFO 1
2:0	Select Port Function (refer to port and function formats)	0 = Select function 0 1 = Select function 1 2 = Select function 2 3 = Select function 3 4-7 = Undefined

Table 6-8 summarizes functions available for each port.

Table 6-8 Port Function Summary

Port	Function 0	Function 1	Function 2	Function 3
A	Flash	SDRAM	N/A	GPIO
B	Flash	SDRAM + Flash + Clock	GPIO	N/A
C	GPIO	MII	N/A	N/A
D	GPIO	MII	N/A	N/A
E	GPIO	SerDes	MII	N/A
F	GPIO	SerDes	MII	N/A
G	GPIO	N/A	N/A	N/A
H	GPIO	MII	GPIO, Clock	N/A

6.6.2 Port Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31	Reserved		TX FIFO Reset	Writing a 1 resets the TX FIFO.	Reserved	Reserved
30			RX FIFO Reset	Writing a 1 resets the current RX FIFO.		
29	TX Level	Transmit FIFO occupancy	Reserved			
28:24			Output Sets	Refer to port and function formats.		
23:22	Reserved					
21:16	RX Level	Receive FIFO occupancy				

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
15	TX FIFO Underflow Interrupt	FIFO empty during transmit.	Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
14	TX FIFO Watermark Interrupt	Transmit FIFO level equal to or below transmit FIFO watermark trigger level.				
13	RX FIFO Overflow Interrupt	FIFO full during receive.				
12	RX FIFO Watermark Interrupt	Receive FIFO level greater than or equal to Receive FIFO watermark trigger level.				
11:0	Function Interrupt	Refer to port and function formats.				

6.6.3 Port Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	TX_FIFO_DEPTH	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	RX_FIFO_DEPTH	Receive FIFO depth (in 32-bit words)
15:0	Function status	Refer to port and function formats

In many cases, the register definitions depend on the function chosen for a port by the value in its Function Register. For each port, the common register definitions are presented first, followed by the definitions that are specific to each port function.

6.7 Port A Registers

6.7.1 Port A Function Select

Port A includes support for the external flash memory controller. Once program code has been downloaded into the on-chip RAM, and when not accessing the flash memory, Port A pins can be used for GPIO functionality.

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = Flash 1 = SDRAM 2 = Undefined 3 = GPIO 4–7 = Undefined

6.7.2 Port A Flash Function

6.7.2.1 Port A Flash Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:15	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			XFL_START	Writing a 1 causes the controller to execute the command stored in XFL_RW_CADQ.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:9	Reserved					
8	XFL_ERR	Controller is in an error state and must be reset before it can continue.				
7:1	Reserved					
0	XFL_DONE	Controller has completed the user command successfully.				

6.7.2.2 Port A Flash Function Control 0

Bits	Field Name	Description
31:9	Reserved	
8	XFL_EN	Enable direct low-level flash access through the port. Must be enabled to execute XFL_START commands.
7:6	XFL_TWAIT[1:0]	Number of wait states to insert between $\overline{\text{FCE}}$ low and $\overline{\text{WE}}$/$\overline{\text{OE}}$ low.
5:0	XFL_TAVAV[5:0]	Read/write access time expressed in core clock cycles. Changes to XFL_TAVAV[5:0] during the first XFL_TWAIT[1:0] cycles of the external flash device access takes effect on this access. Changes outside this brief period take effect on the next XFL_START pulse or for the next IREAD instruction. **XFL_TAVAV** **Cycles** 0 64 1 0 2... value–1

6.7.2.3 Port A Flash Function Status 0

Bits	Field Name	Description
31:1	Reserved	
0	XFL_ACT	Indicates that the controller is active.

6.7.2.4 Port A Flash RX FIFO

Bits	Field Name	Description
31:24	XFL_READ_DATA[7:0]	Byte of data from the flash device.
23:0	Reserved	

6.7.2.5 Port A Flash TX FIFO

Bits	Field Name	Description
31:0	XFL_RW_CADQ[31:0]	User command, plus address and data.

6.7.3 Port A SDRAM Function

6.7.3.1 Port A SDRAM GPIO Control

Bits	Field Name	Description
31:16	Data[31:16] enable	The data pins are enabled through a combination of the GPIO Control registers and an internal SDRAM controller signal. When the GPIO Control register is programmed to all 1s, the SDRAM controller controls the state of the output driver as necessary for data transfer with the external SDRAM devices.
15	DQM	
13:14	BA[1:0]	
12:0	ADDR[12:0] enable	

6.8 Port B Registers

6.8.1 Port B Function Select

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1.
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = Flash 1 = SDRAM plus Flash 2 = GPIO 3–7 = Undefined

6.8.2 Port B SDRAM Function

6.8.2.1 Port B SDRAM Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	**Description**	**Field**	**Description**	**Descr**	**Descr**
31:15	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			SD_START	Writing a 1 causes the SDRAM controller to execute the command in SD_CMD register.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:9	Reserved					
8	PB6 Interrupt					
7:2	Reserved					
1	SD_ERROR	The controller is in an error state and must be reset.				
0	SD_DONE	The controller has completed the current SD_CMD command.				

6.8.2.2 Port B SDRAM Function Control 0

Bits	Field Name	Description
31:30	SD_CL[1:0]	CAS Latency expressed in number of SD_CLK cycles.
29:28	SD_TRDL[1:0]	Last data-in to PRE.A expressed in SD_CLK cycles, with any fractions rounded up.
27:25	SD_TRCD[2:0]	RAS to CAS Delay expressed in number of SD_CLK cycles.
24:22	SD_TRP[2:0]	Row Precharge Time expressed in number of SD_CLK cycles.
21:18	SD_TRAS[3:0]	Min time from ACT to PRE.A command expressed in SD_CLK cycles.
17:14	SD_TRFC[3:0]	Refresh Cycle Time expressed in SD_CLK cycles.
13:1	SD_REF[15:3]	Refresh Period. Frequency with which auto-refresh commands must be sent to SDRAM, expressed in SD_CLK cycles.
0	EN_REF	Enable Auto Refresh.

6.8.2.3 Port B SDRAM Function Control 1

Bits	Field Name	Description
31:27	Reserved	
26:2	SD_ADDR[26:2]	Word address at which to perform read/write on external SDRAM. Must be word-aligned.
1:0	Reserved	

6.8.2.4 Port B SDRAM Function Control 2

Bits	Field Name	Description
31	Reserved	
30:24	SD_BURST_SIZE [6:0]	Length for SDRAM read and write accesses expressed in number of words.
23	Reserved	
22:20	SD_CMD[2:0]	SDRAM controller command. Refer to Table 5-10 for values.
19:10	Reserved	
9:8	PB6 Int Cfg	
7:0	CLK_DIV[7:0]	Clock output divider

6.8.2.5 Port B SDRAM Function Status 0

Bits	Field Name	Description
31:16	Reserved	
15:8	PHASE[7:0]	Clock output phase
7:1	Reserved	
0	SD_ACT	Indicates that the SDRAM controller is busy processing a user command.

6.8.2.6 Port B SDRAM Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.8.2.7 Port B SDRAM RX FIFO

Bits	Field Name	Description
31:0	SD_RD_DATA[31:0]	Read data

6.8.2.8 Port B SDRAM TX FIFO

Bits	Field Name	Description
31:0	SD_WR_DATA[31:0]	Write data

6.9 Port C Registers

6.9.1 Port C Function Select

Bits	Field Name	Description
31:30	unused	Refer to Section 6.6.1
29:24	TX Set	
23:22	unused	
21:16	RX Set	
15:8	unused	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = Select GPIO 1 = Select MII 2–7 Undefined

6.9.2 Port C MII Function

6.9.2.1 Port C MII Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:17	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			TX_START	Writing a 1 starts transmission.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11	Reserved					
10	PC13 Interrupt					
9	PC12 Interrupt					
8	THRESHOLD_INT	The frame currently being received has reached the threshold at which it will not be discarded (32 bytes).				
7	RX_EOP	End-of-packet detected during receive.				
6	RX_SFD	Start of Frame delimiter detected during receive.				
5	RX_ERR	Error detected during receive.				
4	TX_EOP	End of transmission.				
3	COL	Collision detected during transmission.				
2	CRS	Carrier sense.				
1	ODD_NIB_ERR	Odd nibble reception error.				
0	FALSE_CARRIER	False carrier message.				

6.9.2.2 Port C MII Function Control 0

Bits	Field Name	Description
31:24	CLK_DIV[7:0]	Clock divisor for phy-mode.
23:19	Reserved	
18	REVERSE_MII	Set the MII controller into physical-side mode.
17	HALF_DUPLEX	Set the MII controller into half duplex mode.
16	RX_EN	Enable the receiver.
15:0	TX_BYTE_COUNT	Count in bytes of the amount of data to be transmitted.

6.9.2.3 Port C MII Function Control 1

register offset:

Bits	Field Name	Description
31:0	Reserved	

6.9.2.4 Port C MII Function Control 2

Bits	Field Name	Description
31:30	PC13 Int Cfg	
29:28	PC12 Int Cfg	
27:0	Reserved	

6.9.2.5 Port C MII Function Status 0

Bits	Field Name	Description
31:20	Reserved	
19	CRC_OK	The CRC sent with the most recently received packet matches the CRC calculated on the payload of the same packet.
18	RX_FIFO_SELECT	Indicates to which FIFO the currently received frame is being written. 0 = FIFO 0 1 = FIFO 1
17	COLLISION	The state of the COL signal after being synchronized to the core clock domain.
16	CARRIER_SENSE	The state of the CRS signal after being synchronized to the core clock domain.
15:0	RX_BYTE_COUNT	Total number of bytes received, including the CRC, but not including the SFD or any part of the preamble.

6.9.2.6 Port C MII Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.9.2.7 Port C MII RX FIFO

Bits	Field Name	Description
31:24	RX Byte[0]	
23:16	RX Byte[1]	
15:8	RX Byte[2]	
7:0	RX Byte[3]	

6.9.2.8 Port C MII TX FIFO

Bits	Field Name	Description
31:24	TX Byte[0]	
23:16	TX Byte[1]	
15:8	TX Byte[2]	
7:0	TX Byte[3]	

6.10 Port D Registers

6.10.1 Port D Function Select

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = GPIO 1 = MII 2–7 = Undefined

6.10.2 Port D MII Function

6.10.2.1 Port D MII Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:17	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			TX_START	Writing a 1 to this bit starts transmission.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:10	Reserved					
9	PD16 Interrupt					
8	THRESHOLD_INT	The frame currently being received has reached the threshold at which it will not be discarded (32 bytes).				
7	RX_EOP	End-of-packet detected during receive.				
6	RX_SFD	Start of Frame delimiter detected during receive.				
5	RX_ER	Error detected during receive.				
4	TX_EOP	End of transmission.				
3	COL	Collision detected during transmission.				
2	CRS	Carrier sense.				
1	ODD_NIB_ERR	Odd nibble reception error.				
0	FALSE_CARRIER	False carrier message.				

6.10.2.2 Port D MII Function Control 0

Bits	Field Name	Description
31:24	CLK_DIV[7:0]	Clock divisor for phy-mode
23:19	Reserved	
18	REVERSE_MII	Set the MII controller into physical-side mode.
17	HALF_DUPLEX	Set the MII controller into half duplex mode.
16	RX_EN	Enable the receiver.
15:0	TX_BYTE_COUNT	Count in bytes of the amount of data to be transmitted.

6.10.2.3 Port D MII Function Control 1

Bits	Field Name	Description
31:0	Reserved	

6.10.2.4 Port D MII Function Control 2

Bits	Field Name	Description
31:30	PD16 Int Cfg	
29:0	Reserved	

6.10.2.5 Port D MII Function Status 0

Bits	Field Name	Description
31:20	Reserved	
19	CRC_OK	The CRC sent with the most recently received packet matches the CRC calculated on the payload of the same packet.
18	RX_FIFO_SELECT	Indicates to which FIFO the currently received frame is being written. 0 = FIFO 0 1 = FIFO 1
17	COLLISION	The state of the COL signal after being synchronized to the core clock domain.
16	CARRIER_SENSE	The state of the CRS signal after being synchronized to the core clock domain.
15:0	RX_BYTE_COUNT	Total number of bytes received, including the CRC, but not including the SFD or any part of the preamble.

6.10.2.6 Port D MII Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.10.2.7 Port D MII RX FIFO

Bits	Field Name	Description
31:24	RX BYTE[0]	
23:16	RX BYTE[1]	
15:8	RX BYTE[2]	
7:0	RX BYTE[3]	

6.10.2.8 Port D MII TX FIFO

Bits	Field Name	Description
31:24	TX BYTE[0]	
23:16	TX BYTE[1]	
15:8	TX BYTE[2]	
7:0	TX BYTE[3]	

6.11 Port E Registers

6.11.1 Port E Function Select

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = GPIO 1 = SerDes 2 = MII 3 = GPSI 4–7 = Undefined

6.11.2 Port E SerDes Function

6.11.2.1 Port E SerDes Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:15	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			TXBUF_VALID	TX data is valid.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:10	Reserved					
9	PE3 Int					
8	Reserved					
7	RXERR	USB: SerDes has detected 7 consecutive ones.				
6	RXEOP	USB: Asserted at end of packet. GPSI: Asserted at deassertion of RxEn.				
5	SYND	USB: Received data matches sync pattern (RSYNC).				
4	TXBE	SerDes has consumed data present at TXBUF.				
3	TXEOP	SerDes has completed transmitting all available data and no new data is available. In 10Base-T or USB modes, an EOP is transmitted after the last data.				
2	SX LP	USB: Bus idle after SerDes stops driving the bus.				
1	RXBF	Receive data is available.				
0	RXXCRS	USB: RxBusy is detected.				

6.11.2.2 Port E SerDes Function Control 0

Bits	Field Name	Description
31	GLOBAL_EN	0: Disable SerDes output (without resetting device). SerDes I/O pins become accessible through GPIO. 1: SerDes I/O pins are driven from SerDes as defined through the mode register.
30	LOOP_BACK,	Loopback control 0: disable loopback 1: enable loopback
29	TX_DATA_INV	Invert all transmitted data.
28:24	TXSCNT[4:0]	Transmit bit count.
23:16	MODE[7:0]	SerDes mode/submode select. 23:20 – PRS 19:18 – SUBM 17:16 – unused
15:0	CLKDIV[15:0]	Clock divider for generating serial I/O clock from I/O clock.

6.11.2.3 Port E SerDes Function Control 1

Bits	Field Name	Description
31:24	SYNCMASK[7:0]	Mask for RSYNC
23:16	RSYNC[7:0]	USB: Sync pattern
15:10	Reserved	
9	BIT_ORDER	Serialization order: 0: LSB first 1: MSB first
8	Reserved	
7	SPI_MASTER_SEL	SPI or GPSI only: 0: Slave 1: Master
6	USB_SYNC_IGNORE	USB only: 1 = Do not detect sync.
5	REV_POLARITY_EN	1 = Invert received data.
4:0	RXSCNT[4:0]	Receive count interrupt level

6.11.2.4 Port E SerDes Function Control 2

Bits	Field Name	Description
31:30	PE3 Int Cfg	
29:24	Reserved	
23:16	Reserved	
15:0	TXBUF[15:0]	Data for transmit operations

6.11.2.5 Port E SerDes Function Status 0

Bits	Field Name	Description
31:10	Reserved	
9	LINK_POLARITY	10Base-T only: Link pulse detected with reverse polarity.
8	CRS_STATUS	10Base-T only: Current state of carrier.
7:5	Reserved	
4:0	RXCTR[4:0]	Actual number of received bits. Exceptions for the last transfer are: – Shows number of bits if less than 8. – Shows 8 if less than 16 and greater than 8. – Shows 16 if received count is ≥ 16 and rxscnt = 16.

6.11.2.6 Port E SerDes Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.11.2.7 Port E SerDes RX FIFO

Bits	Field Name	Description
31:16	Reserved	
15:0	RX DATA	Read data

6.11.2.8 Port E SerDes TX FIFO

Bits	Field Name	Description
31:0	Reserved	

6.11.3 Port E MII Function

When the MII function is selected for Port E, Port F must also select the MII function.

6.11.3.1 Port E MII Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:17	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			TX_START	Writing a 1 to this bit starts transmission.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:10	Reserved					
9	PE3 Interrupt					
8	THRESHOLD_INT	The frame currently being received has reached the threshold at which it will not be discarded (32 bytes).				
7	RX_EOP	End-of-packet detected during receive.				
6	RX_SFD	Start of Frame delimiter detected during receive.				
5	RX_ER	Error detected during receive.				
4	TX_EOP	End of transmission.				
3	COL	Collision detected during transmission.				
2	CRS	Carrier sense.				
1	ODD_NIB_ERR	Odd nibble reception error.				
0	FALSE_CARRIER	False carrier message.				

6.11.3.2 Port E MII Function Control 0

Bits	Field Name	Description
31:24	CLK_DIV[7:0]	Clock divisor for phy-mode
23:19	Reserved	
18	REVERSE_MII	Set the MII controller into physical-side mode.
17	HALF_DUPLEX	Set the MII controller into half duplex mode.
16	RX_EN	Enable the receiver.
15:0	TX_BYTE_COUNT	Count in bytes of the amount of data to be transmitted.

6.11.3.3 Port E MII Function Control 1

Bits	Field Name	Description
31:0	Reserved	

6.11.3.4 Port E MII Function Control 2

Bits	Field Name	Description
31:30	PE3 Int Cfg	
29:0	Reserved	

6.11.3.5 Port E MII Function Status 0

Bits	Field Name	Description
31:20	Reserved	
19	CRC_OK	The CRC sent with the most recently received packet matches the CRC calculated on the payload of the same packet.
18	RX_FIFO_SELECT	Indicates to which FIFO the currently received frame is being written. 0 = FIFO 0 1 = FIFO 1
17	COLLISION	The state of the COL signal after being synchronized to the core clock domain.
16	CARRIER_SENSE	The state of the CRS signal after being synchronized to the core clock domain.
15:0	RX_BYTE_COUNT	Total number of bytes received, including the CRC, but not including the SFD or any part of the preamble.

6.11.3.6 Port E MII Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.11.3.7 Port E MII RX FIFO

Bits	Field Name	Description
31:24	RX BYTE[0]	
23:16	RX BYTE[1]	
15:8	RX BYTE[2]	
7:0	RX BYTE[3]	

6.11.3.8 Port E MII TX FIFO

Bits	Field Name	Description
31:24	TX BYTE[0]	
23:16	TX BYTE[1]	
15:8	TX BYTE[2]	
7:0	TX BYTE[3]	

6.11.4 Port E High-Speed GPSI Function

Port E can be used as a high-speed GPSI port. This should not be confused with the SerDes-based GPSI port referenced in Section 6.11.2 and discussed in Section 5.5.8.

6.11.4.1 Port E GPSI Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	**Description**	**Field**	**Description**	**Descr**	**Descr**
31	FIFO Control Refer to Section 6.6.2		TX FIFO Reset	Writing a 1 to this bit resets the TX FIFO.	Reserved	Reserved
30			RX FIFO Reset	Writing a 1 to this bit resets the RX FIFO.		
29:18			Reserved			
17			TX_HALT	Writing a 1 to this bit stops transmission.		
16			TX_START	Writing a 1 to this bit starts transmission.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:5	Reserved					
4	RX_SFD	SFD detected.				
3	TX_EOP	End of transmission.				
2	RX_EOP	End of Packet detected during receive.				
1	COL	Collision detected during transmission.				
0	CRS	Carrier sense.				

6.11.4.2 Port E GPSI Function Control 0

Bits	Field Name	Description
31:16	TX_BIT_CNT	Number of bits to be transmitted.
15:8	TX_CLOCK_DIV	Transmit Clock Divisor, used in Master mode.
7:2	Reserved	
1	TX_CLK_POL	If 1, inverse TX CLK.
0	MASTER	If 1, GPSI is Master and drives TX and RX clocks.

6.11.4.3 Port E GPSI Function Control 1

Bits	Field Name	Description
31:16	Reserved	
15:8	RX_CLOCK_DIV	Receive Clock Divisor, used in Master mode.
7:4	Reserved	
3	CRS_POL	If 1, CRS pin is active low.
2	COL_POL	If 1, COL pin is active low.
1	RX_CLK_POL	If 1, Inverse RX CLK.
0	RX_ENABLE	Enable the receiver.

6.11.4.4 Port E GPSI Function Control 2

Bits	Field Name	Description
31:16	SFD_PATTERN	The pattern to be matched to detect the Start of Frame delimiter
15:0	SFD_MASK	A 1 in a given bit position allows the corresponding bit in SFD_PATTERN to be used in SFD compare.

6.11.4.5 Port E GPSI Function Status 0

Bits	Field Name	Description
31:18	Reserved	
17	COLLISION	The state of the COL signal after being synchronized to the core clock domain.
16	CARRIER_SENSE	The state of the CRS or TxBUSY signal after being synchronized to the core clock domain.
15:0	RX_BIT_CNT	Number of bits received. The code in bits 4:0 shows the number of bits within the 32-bit word, starting from bit 0.

6.11.4.6 Port E GPSI Function Status 1

Bits	Field Name	Description
31:0	Reserved	

6.11.4.7 Port E GPSI RX FIFO

Bits	Field Name	Description
31:24	RX BYTE[0]	
23:16	RX BYTE[1]	
15:8	RX BYTE[2]	
7:0	RX BYTE[3]	

6.11.4.8 Port E GPSI TX FIFO

Bits	Field Name	Description
31:24	TX BYTE[0]	
23:16	TX BYTE[1]	
15:8	TX BYTE[2]	
7:0	TX BYTE[3]	

6.12 Port F Registers

6.12.1 Port F Function Select

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = GPIO 1 = SerDes 2 = MII 3–7 = Undefined

6.12.2 Port F SerDes Function

6.12.2.1 Port F SerDes Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:15	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			TXBUF_VALID	TX data is valid.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11:10	Reserved					
9	PF3 Int					
8	Reserved					
7	RXERR	10Base-T: Manchester data phase error is detected. USB: SerDes has detected 7 consecutive ones.				
6	RXEOP	10Base-T and USB: Asserted at end of packet. GPSI: Asserted at deassertion of RxEn.				
5	SYND	10Base-T and USB: Received data matches sync pattern (RSYNC).				
4	TXBE	SerDes has consumed data present at TXBUF.				
3	TXEOP	SerDes has completed transmitting all available data and no new data is available. In 10Base-T or USB modes, an EOP is transmitted after the last data.				
2	SX LP	10Base-T: Link pulse detected USB: Bus idle after SerDes stops driving the bus.				
1	RXBF	Receive data is available.				
0	RXXCRS	10Base-T: Carrier sense gained or lost as per CRS_INT_POLARITY USB: RxBusy is detected.				

6.12.2.2 Port F SerDes Function Control 0

Bits	Field Name	Description
31	GLOBAL_EN	0: Disable SerDes output (without resetting device). SerDes I/O pins become accessible through GPIO 1: SerDes I/O pins are driven from SerDes as defined through the mode register.
30	LOOP_BACK,	Loop-back control 0: disable loopback 1: enable loopback
29	TX_DATA_INV	Invert all transmitted data.
28:24	TXSCNT[4:0]	Transmit bit count.
23:16	MODE[7:0]	SerDes mode/submode select. 23:20 – PRS 19:18 – SUBM 17:16 – unused
15:0	CLKDIV[15:0]	Clock divider for generating serial I/O clock from I/O clock.

6.12.2.3 Port F SerDes Function Control 1

Bits	Field Name	Description
31:24	SYNCMASK[7:0]	Mask for RSYNC.
23:16	RSYNC[7:0]	USB: Sync pattern 10Base-T: 17: Squelch enable 16: SW handles dribble bit
15:10	Reserved	
9	BIT_ORDER	Serialization order: 0: LSB first 1: MSB first
8	CRS_INT_POLARITY	10Base-T: 0: RXXCRS interrupt on gain of carrier 1: RXXCRS interrupt on loss of carrier
7	SPI_MASTER_SEL	SPI or GPSI only: 0: Slave 1: Master
6	USB_SYNC_IGNORE	USB only: 1 = Do not detect sync.
5	REV_POLARITY_EN	1 = Invert received data.
4:0	RXSCNT[4:0]	Receive count interrupt level

6.12.2.4 Port F SerDes Function Control 2

Bits	Field Name	Description
31:30	PF3 Int Cfg	
29:24	Reserved	
23:16	SQUELCH TRIM [7:0]	10Base-T: Squelch trim value; set to 0xFF when SerDes 10Base-T is selected.
15:0	TXBUF[15:0]	Data for transmit operations

6.12.2.5 Port F SerDes Function Status 0

Bits	Field Name	Description
31:10	Reserved	
9	LINK_POLARITY	10Base-T only: Link pulse detected with reverse polarity.
8	CRS_STATUS	10Base-T only: Current state of carrier
7:5	Reserved	
4:0	RXCTR[4:0]	Actual number of received bits. Exceptions for the last transfer are: – Shows number of bits if less than 8. – Shows 8 if less than 16 and greater than 8. – Shows 16 if received count is ≥ 16 and rxscnt = 16.

6.12.2.6 Port F SerDes Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.12.2.7 Port F SerDes RX FIFO

Bits	Field Name	Description
31:16	Reserved	
15:0	RX DATA	Read data

6.12.2.8 Port F SerDes TX FIFO

Bits	Field Name	Description
31:0	Reserved	

6.12.3 Port F MII Function

Port E and Port F together support a single MII interface controller. When the MII function is selected for Port E, Port F must also select the MII function. In that case, the registers of Port E control the MII function and the remaining registers of Port F are not used.

6.13 Port G Registers

6.13.1 Port G Function Select

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = GPIO 1–7 = Undefined

6.14 Port H Registers

6.14.1 Port H Function Select

Bits	Field Name	Description
31:30	reserved	Refer to Section 6.6.1
29:24	TX Set	
23:22	reserved	
21:16	RX Set	
15:8	reserved	
7:4	Function Reset	
3	Receive FIFO Select	
2:0	Select	0 = GPIO 1 = MII 2 = Clock 3–7 = Undefined

6.14.2 Port H MII Function

6.14.2.1 Port H MII Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:17	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16			TX_START	Writing a 1 starts transmission.		
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11	Reserved					
10	PH15 Interrupt					
9	PH14 Interrupt					
8	THRESHOLD_INT	The frame currently being received has reached the threshold at which it will not be discarded (32 bytes).				
7	RX_EOP	End-of-packet detected during receive.				
6	RX_SFD	Start of Frame delimiter detected during receive.				
5	RX_ERR	Error detected during receive.				
4	TX_EOP	End of transmission.				
3	COL	Collision detected during transmission.				
2	CRS	Carrier sense.				
1	ODD_NIB_ERR	Odd nibble reception error.				
0	FALSE_CARRIER	False carrier message.				

6.14.2.2 Port H MII Function Control 0

Bits	Field Name	Description
31:24	CLK_DIV[7:0]	Clock divisor for phy-mode.
23:19	Reserved	
18	REVERSE_MII	Set the MII controller into physical-side mode.
17	HALF_DUPLEX	Set the MII controller into half duplex mode.
16	RX_EN	Enable the receiver.
15:0	TX_BYTE_COUNT	Count in bytes of the amount of data to be transmitted.

6.14.2.3 Port H MII Function Control 1

register offset:

Bits	Field Name	Description
31:0	Reserved	

6.14.2.4 Port H MII Function Control 2

Bits	Field Name	Description
31:30	PH15 Int Cfg	
29:28	PH14 Int Cfg	
27:0	Reserved	

6.14.2.5 Port H MII Function Status 0

Bits	Field Name	Description
31:20	Reserved	
19	CRC_OK	The CRC sent with the most recently received packet matches the CRC calculated on the payload of the same packet.
18	RX_FIFO_SELECT	Indicates to which FIFO the currently received frame is being written. 0 = FIFO 0 1 = FIFO 1
17	COLLISION	The state of the COL signal after being synchronized to the core clock domain.
16	CARRIER_SENSE	The state of the CRS signal after being synchronized to the core clock domain.
15:0	RX_BYTE_COUNT	Total number of bytes received, including the CRC, but not including the SFD or any part of the preamble.

6.14.2.6 Port H MII Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

6.14.2.7 Port H MII RX FIFO

Bits	Field Name	Description
31:24	RX Byte[0]	
23:16	RX Byte[1]	
15:8	RX Byte[2]	
7:0	RX Byte[3]	

6.14.2.8 Port H MII TX FIFO

Bits	Field Name	Description
31:24	TX Byte[0]	
23:16	TX Byte[1]	
15:8	TX Byte[2]	
7:0	TX Byte[3]	

6.14.3 Port H Clock Function

6.14.3.1 Port H Clock Interrupt Registers

Bits	Interrupt Status Register		Interrupt Set Register		Interrupt Mask Reg	Interrupt Clear Reg
	Field	Description	Field	Description	Descr	Descr
31:15	FIFO Control Refer to Section 6.6.2		FIFO Control Refer to Section 6.6.2		Reserved	Reserved
16						
15:12			Interrupt Set	Writing a 1 to any of bits 0–15 sets the corresponding interrupt status bit.	A 1 in any of bits 0–15 enables the corresponding interrupt.	Writing a 1 to any of bits 0–15 clears the corresponding interrupt status bit.
11	Reserved					
10	PH15 Interrupt					
9	PH14 Interrupt					
8:0	Reserved					

6.14.3.2 Port H Clock Function Control 0

Bits	Field Name	Description
31:0	Reserved	

6.14.3.3 Port H Clock Function Control 1

Bits	Field Name	Description
31:0	Reserved	

6.14.3.4 Port H Clock Function Control 2

Bits	Field Name	Description
31:30	PH15 Int Cfg	
29:28	PH14 Int Cfg	
27:8	Reserved	
7:0	CLK_DIV[7:0]	Clock output divider

6.14.3.5 Port H Clock Function Status 0

Bits	Field Name	Description
31:8	Reserved	
7:0	PHASE[7:0]	Clock output phase

6.14.3.6 Port H Clock Function Status 1

Bits	Field Name	Description
31:30	Reserved	
29:24	tx_fifo_depth	Transmit FIFO depth (in 32-bit words)
23:22	Reserved	
21:16	rx_fifo_depth	Receive FIFO depth (in 32-bit words)
15:0	Reserved	

7.0 Electrical Specifications

7.1 Absolute Maximum Ratings

Absolute maximum ratings, beyond which permanent damage may occur. Correct operation not guaranteed outside of DC specifications listed Section 7.2.

Parameter	Min	Typical	Max	Units	Conditions
Ambient temperature under bias	–40		125	°C	
Storage temperature	–65		150	°C	
Voltage on DVDD, PLL1Vdd, PLL2Vdd, A1Vdd, A2Vdd with respect to Vss	0		1.5	V	
Voltage on IOVdd with respect to Vss	0		4.5	V	
Input voltage on Port A-H pins, $\overline{\text{RST}}$, $\overline{\text{TSS}}$, TSCK, and TSI pins	–0.3		5.7	V	
Input voltage on PFRDN and PFRDP pins	–0.5		1.5	V	clamps at < –0.5V and > 1.5V
Output voltage on Port A-H pins, TSO pin			4.5	V	
Max allowable sink/source current for group of pins between IOVdd pins				mA	
Junction Temperature, Tj			125	°C	Note A
Thermal resistance of the package between the die and ambient air, θ_{JA}		34		°C/W	Airflow = 0 m/sec
		31		°C/W	Airflow = 1 m/sec
		30		°C/W	Airflow = 2 m/sec
Thermal resistance of the package between the die and the case of the package, θ_{JC}		11		°C/W	

Note A: The junction temperature of the die is calculated with Pd = (Tj - Ta)/θ_{JA} , where
Pd = Power dissipated by the die
Tj = Junction temperature of the die
Ta = ambient temperature of the surrounding air
θ_{JA} = thermal resistance of the package between the die and ambient air.

For example, maximum allowable power dissipated by the die at an ambient air temperature of 85°C with zero airflow = (125 - 85)/34 = 1.18W. For an IP3023 in a typical Router/Gateway application, max Pd = (0.36A*1.32V) + (0.03A* 3.6V) = 0.6W, which in this example is less than 1.18W, so Tj < 125°C. Minimum airflow, maximum Idd and Vdd for each supply rail, and maximum ambient temperature need to be determined for each application, to ensure that Tj < 125°C.

7.2 DC Specifications

Operating Ambient Air temperature = -40°C to +85°C, except where noted otherwise.

Symbol	Parameter	Min	Typical	Max	Units	Conditions	Part Number
DVDD = PLL1VDD = PLL2VDD = A1VDD = A2VDD	Supply Voltages (Digital, PLL, Analog). All 5 should be at the same voltage, but should have filters between the 5 supplies. Note 5.	1.14	1.25	1.32	V	0 to 70°C, 250MHz max	IP3023/BG228-250
		1.17	1.25	1.32	V	-40 to +85°C, 250MHz max	
		1.30	1.37	1.40	V	0 to 70°C, 325MHz max	IP3023/BG228-325
IOVDD	I/O Supply Voltage	3.0	3.3	3.6	V	DVDD should never exceed IOVDD by > 0.4V (even during power up/down). Note 4	
Iddmax	Supply Current, Worst case software: DVDD + PLL1VDD + PLL2VDD + A1VDD + A2VDD		550	690	mA	VDD = 1.2V for Typ, 1.32V for Max, Core = Continuous 250MHz, no power savings modes	
Iddtyp	Typical Idd: DVDD + PLL1VDD + PLL2VDD + A1VDD + A2VDD		350		mA	Fcore = 250MHz, VDD = 1.2V.	
			235		mA	Fcore = 120MHz, VDD = 1.2V.	
			25		mA	Fcore = 10MHz, VDD = 1.2V.	
IddIO	Supply Current, Active: IOVDD		25		mA	IOVDD = 3.3V	
Iddmin	Supply Current, Low Power mode (minimum core clock freq.): DVDD + PLL1VDD + PLL2VDD + A1VDD + A2VDD. DVDD = 1.2V for Typ, 1.32V for Max, OSC_IN = 10MHz, core frequency = 312.5KHz.			36	mA	PLLs on	
				27	mA	PLLs powered down	
Iminio	Supply Current, Low Power mode: IOVDD supply			0.5	mA	IOVDD = 3.6V, no loads, no floating inputs, no switching I/Os.	
Vih	Input high voltage: all Port Pins (Ports A-H), $\overline{\text{TSS}}$, TSCK, TSI, $\overline{\text{TEST}}$0, TEST1, TEST2, and $\overline{\text{RST}}$ pins	2.0		5.5	V		
Vil	Input low voltage: all Port Pins (Ports A-H), $\overline{\text{TSS}}$, TSCK, TSI, $\overline{\text{TEST}}$0, TEST1, TEST2, and $\overline{\text{RST}}$ pins			0.8	V		
Vina	Analog input voltage: PFRDN and PFRDP pins	0		A1Vdd	V		

Symbol	Parameter	Min	Typical	Max	Units	Conditions	Part Number
Ileak	Input leakage current: all Port Pins, and TSO pin	–10		10	µA	Inputs held at 0V or 5.5V. TSO measured while $\overline{\text{TSS}}$ = high.	
Ileak	Input leakage current: $\overline{\text{RST}}$, $\overline{\text{TSS}}$, and TSI pins	–110		–25	µA	Inputs held at 0V. Note 2.	
Ileak	Input leakage current: $\overline{\text{RST}}$, $\overline{\text{TSS}}$, and TSI pins			10	µA	Inputs held at 5.5V. Note 2.	

Symbol	Parameter	Min	Typical	Max	Units	Conditions
Ileak	Input leakage current: TSCK, TEST0, TEST1, TEST2 pins	–10			µA	Inputs held at 0V. Note 3.
Ileak	Input leakage current: TSCK, TEST0, TEST1, TEST2 pins	10		110	µA	Inputs held at 5.5V. Note 3.
Ileak	Input leakage current: OSC_IN, PFRDN, PFRDP pins	–15		15	µA	Input held at 0V or A1Vdd, squelch disabled
Ioh	Output high current: TSO pin, and Ports A-H, except PB6, PF5, PF6 pins.	8	24	-	mA	Voh = 2.4V, IOVdd = 3.0V to 3.6V
	Output high current: PB6 pin.	14	48	-	mA	
	Output high current: PF5, PF6 pins.	24	65	-	mA	
Iol	Output low current: TSO pin, and Ports A-H, except PB6, PF5, PF6 pins.	6	11	-	mA	Vol = 0.4V, IOVdd = 3.0V to 3.6V
	Output low current: PB6 pin.	14	22	-	mA	
	Output low current: PF5, PF6 pins.	16	30	-	mA	

Note 1: Data in the Typical column is at 1.2/3.3V, 25°C, unless otherwise stated.

Note 2: These pins have an internal active pullup to a threshold drop below IOVDD (39kΩ min, 55kΩ typ, 85kΩ max).

Note 3: These pins have an internal active pulldown to a threshold drop above IOVSS (45kΩ min, 93kΩ typ, 198kΩ max).

Note 4: During the time after IOVDD is powered up, the I/Os will be tri-stated even if DVDD is not powered up.

Note 5: The designer must ensure that voltage on the pins of the device is always above the minimum operation voltage. The typical voltage regulator output voltage should be above the midpoint between the above min and max voltages, to allow for voltage drop on the board's traces between the regulator output and the pins of the device, even when the device is running at full speed.

7.3 AC Specifications

Operating Ambient Air temperature = -40°C to 85°C, except where noted otherwise

Protocol	Symbol	Parameter	Min	Typi-cal	Max	Units	Conditions
System	fCORE	Core Frequency. Execution of instructions from Program RAM	0.3125		250	MHz	-40°C to +85°C, part number IP3023/BG228-250
					325	MHz	0°C to 70°C, part number IP3023/BG228-325
	fPLL	PLL (Core & Serial) input frequency	2		20	MHz	At output of the Ref Clock Divider in Figure 3-3.
	fXTAL	Crystal frequency on OSC_IN / OSC_OUT	10		20	MHz	Crystal with ±100ppm rating.
	tVdd	DVdd rise time to ensure Power-On reset			10	ms	Must be > 1V within 10ms of power on when relying on Power-On reset.
		PLL (core and serial) stabilization time			1000	cycles	Cycles into PLL (after reference clock divider)
	tRST	External Reset Pulse	1			µs	
	tRHO	Reset hold off time	50		150	ms	Time after deassertion of reset until internal reset released. Allows for full crystal startup.
		Crystal Startup Time			80	ms	
MII	fMCLK	MII TX_CLK and RX_CLK clock frequency	2.5		25	MHz	Max = 25MHz +200ppm. Core Clock must be faster.
	tMRXS	RXD, RX_DV, RX_ERR setup to RX_CLK rising	10			ns	
	tMRXH	RXD, RX_DV, RX_ERR hold from RX_CLK rising	10			ns	
	tMTXD	TX_ER, TX_EN, TXD from TX_CLK rising			20	ns	
SDRAM	fSD	SDRAM Clock Frequency			66	MHz	Core clock must be faster.
	tSDS	DATA setup to SDRAM_CLK rising	2			ns	
	tSDH	DATA hold from SDRAM_CLK rising	1.5			ns	
	tSDD	DATA, ADDR, Control (RAS, CAS, etc.) from SDRAM_CLK rising	1		12	ns	

Protocol	Symbol	Parameter	Min	Typi-cal	Max	Units	Conditions
SerDes	fPD17	PD17 pin (Auxiliary I/O) clock frequency	10		20	MHz	Note A1
	fIO	I/O PLL output clock frequency (into SerDes divider)	5		250	MHz	Note A1
	fSDCK	SerDes internal (after SerDes divider) clock rate	0.0000762		100	MHz	Core clock must be faster. Min = 5MHz/65536. Note A1
	fSDPCK	SerDes pin Clock Rate	0		25	MHz	This must be at least 4 times slower than tSDCK. Note A1
	tEDS	Input data setup to active edge of clock	7			ns	Note A1
	tEDH	Input data hold from active edge of clock	7			ns	Note A1
	tEDD	Output data from active edge of clock			20	ns	Note A1
GPIO	tGS	Delay from externally applied input signal to first synchronization register			5	ns	Note A2
	tGDD	Delay from internal clock to output pin configured as GPIO			12	ns	Note A2
	tGDE	Enable delay from internal clock to output pin configured as GPIO			12	ns	Note A2
High-Speed GPSI	fTXCLK	TxCLK Frequency			50	MHz	
	fRXCLK	RxCLK Frequency			50	MHz	
	tCQ	TxCLK to signal out	0		15	ns	
	tS	Setup to RxCLK	5			ns	
	tH	Hold from RxCLK	5			ns	

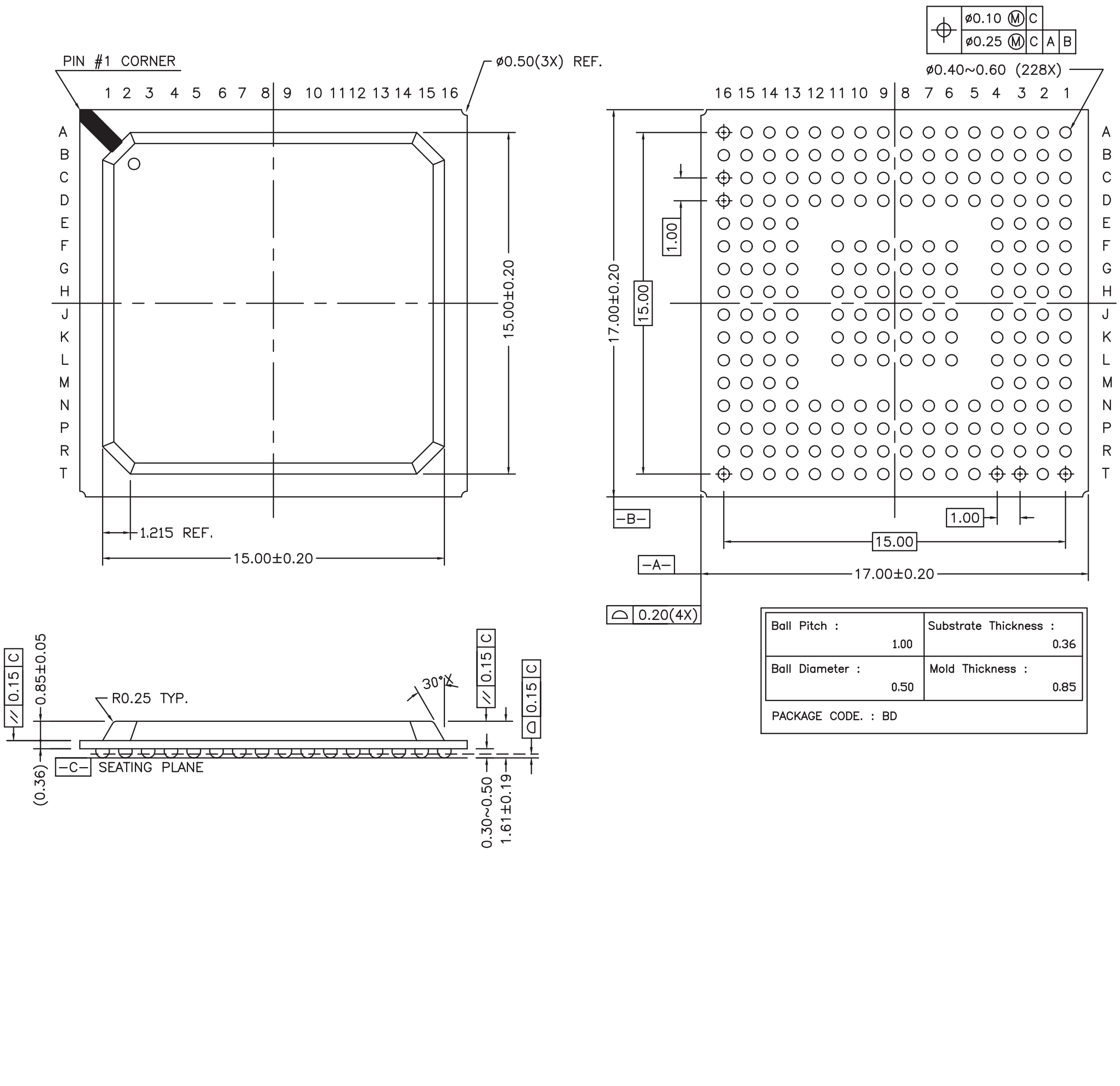
Note A1: The relevant signals and active clock edge for the SerDes depend on the current configuration. These timings are relevant for modes that use an externally supplied clock.

Note A2: GPIO timings are not directly visible to the user. These numbers are useful, however, when performing timing analysis for applications that use direct I/O manipulation.

8.0 Package Dimensions

OLD PACKAGE (datecodes generally before 2006), with datecode of 10DGyywwzz -

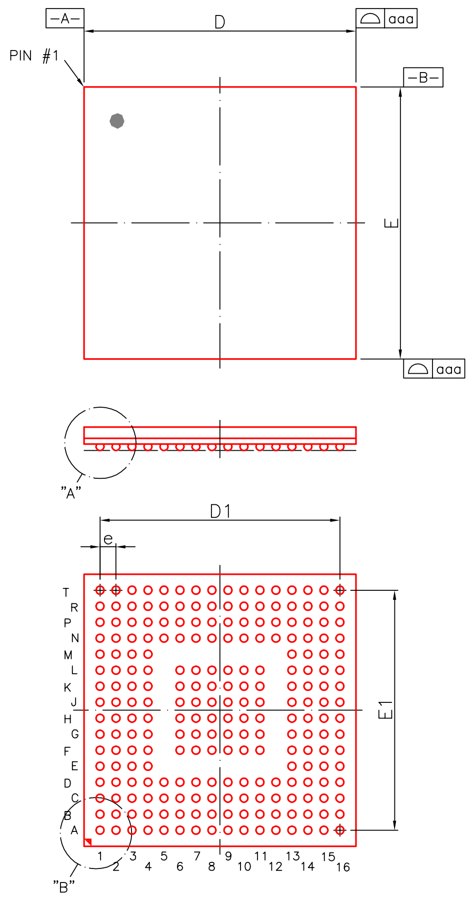
IP3023/BG228-250 only - 228 balls, 17x17x1.61mm package, 1mm ball pitch. All dimensions in mm.

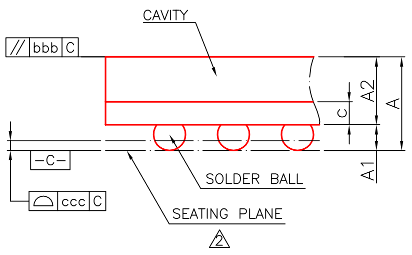


Ball Pitch : 1.00	Substrate Thickness : 0.36
Ball Diameter : 0.50	Mold Thickness : 0.85
PACKAGE CODE. : BD	

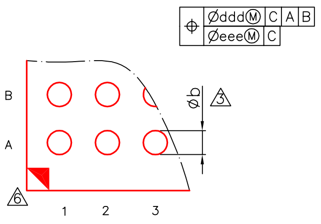
NEW PACKAGE (datecodes generally after 2005), with datecode of 10DHyywwzz -

All IP3023 Products – 228 balls, 17x17x1.56mm package, 1mm ball pitch.





DETAIL : "A"



DETAIL : "B"

Symbol	Dimension in mm			Dimension in inch		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.56	---	---	0.061
A1	0.35	0.40	0.45	0.014	0.016	0.018
A2	1.01	1.06	1.11	0.040	0.042	0.044
c	0.32	0.36	0.40	0.013	0.014	0.016
D	16.90	17.00	17.10	0.665	0.669	0.673
E	16.90	17.00	17.10	0.665	0.669	0.673
D1	---	15.00	---	---	0.591	---
E1	---	15.00	---	---	0.591	---
e	---	1.00	---	---	0.039	---
b	0.45	0.50	0.55	0.018	0.020	0.022
aaa	---	0.10	---	---	0.004	---
bbb	---	0.20	---	---	0.008	---
ccc	---	0.12	---	---	0.005	---
ddd	---	0.15	---	---	0.006	---
eee	---	0.08	---	---	0.003	---
MD/ME	16/16			16/16		

NOTE :

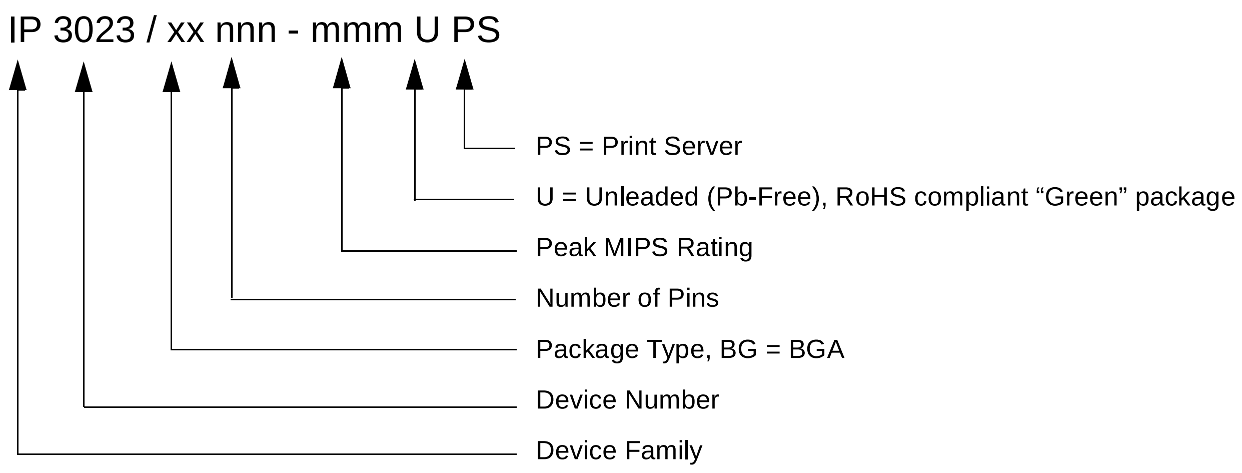
1. CONTROLLING DIMENSION : MILLIMETER.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C.
4. THERE SHALL BE A MINIMUM CLEARANCE OF 0.25mm BETWEEN THE EDGE OF THE SOLDER BALL AND THE BODY EDGE.
5. REFERANCE DOCUMENT : JEDEC MO-216
6. THE PATTERN OF PIN 1 FIDUCIAL IS FOR REFERENCE ONLY.
7. SPECIAL CHARACTERISTICS C CLASS: bbb, ccc

Ball side solder mask ball opening diameter: 0.40 mm.

9.0 Part Numbering

Ordering Information

Part Number	Pins	I/O	Package	Program Flash	Program RAM	Data RAM	Max Core Frequency	Temperature
IP3023/BG228-250	228	138	BGA	up to 4MB external	256KB	64KB	250MHz	-40°C to +85°C
IP3023/BG228-250U	228	138	BGA, Pb-Free, RoHS compliant	up to 4MB external	256KB	64KB	250MHz	-40°C to +85°C
IP3023/BG228-250PS	228	138	BGA	up to 4MB external	256KB	64KB	250MHz	-40°C to +85°C
IP3023/BG228-250UPS	228	138	BGA, Pb-Free, RoHS compliant	up to 4MB external	256KB	64KB	250MHz	-40°C to +85°C
IP3023/BG228-325	228	138	BGA	up to 4MB external	256KB	64KB	325MHz	0°C to +70°C
IP3023/BG228-325U	228	138	BGA, Pb-Free, RoHS compliant	up to 4MB external	256KB	64KB	325MHz	0°C to +70°C



Changes in this Revision

The following table lists the substantial differences from previous versions.

Changed in version	Location	Description of Change
Dec. 9, 2005	Section 7.1	Changed Thermal resistance of the package between the die and ambient air.
	Section 9.0	Added part numbers IP3023/BG228-250PS, IP3023/BG228-250UPS, and IP3023/BG228-325U.
Sept. 7, 2005	Section 9.0	Added diagram of new pin compatible package (old package being obsoleted)
	Section 9.0	Added Unleaded (Pb-Free), RoHS compliant “Green” package part number
	Tables 2-5, 5-3	Added notes - $\overline{\text{FCE}}$ pin should have an external pullup resistor.
	Figure 5-16	Corrected CRS to be connected to IP3023’s TX_EN pin, instead of to VDD.
	Section 5.9.1	Added clarifications on Watchdog timer.
	Section 7.0	Updated sections 7.1, 7.2 notes and formatting
Oct. 28, 2004	Various	Added 325MHz part number - updated sections 1.0, 1.1, 1.3.4, 3.5.4, 3.9, 5.4, and especially sections 7.2, 7.3, and 9.0.
	Section 7.2, 7.3	Changed temperature spec from a range of 0 to 70C, to range of -40 to 85C. Increased max DVDD from 1.26V to 1.32V. Added Note 4.
	Section 7.1	Added max junction temperature, and thermal resistances.
	Table 6-5	Updated rules for PLL power down and clock source select.
	Sections 1.1, 1.37, 3.8.2, 5.9, 5.9.1, 6.2.4, 6.5	Clarified that the “Real Time Timer” and “Watchdog Timer” aren’t really timers, but just compare registers that compare with the Multipurpose Timer to achieve Real Time and Watchdog Timer functions.
	Various	Added register addresses to Table 3-1, 3-2, 6-1.
Aug. 24, 2004	Section 3.10	Crystal ESR values changed. Feedback resistor added.
	Section 6.11.2.3	SPI_MASTER_SEL: Settings for Master and Slave interchanged.
	Section 6.12.2.3	SPI_MASTER_SEL: Settings for Master and Slave interchanged.

Sales and Technical Support Contact Information

For the latest contact and support information on IP devices, please visit the Ubicom Web site at www.ubicom.com. The site contains technical literature, local sales contacts, tech support, and many other features.

The Products are not authorized for use in life support systems or under conditions where failure of the Product would endanger the life or safety of the user, except when prior written approval is obtained from Ubicom, Inc. Ask your sales representive for details.

635 Clyde Avenue
Mountain View, CA 94043

Tel:	650.210.1500
Fax:	650.210.8715
Email:	sales@ubicom.com
Web:	www.ubicom.com

Ubicom, Inc. develops and markets wireless network processor and software platforms that enable all electronic devices to be connected to each other – securely, cost-effectively and transparently. With headquarters in Mountain View, California, Ubicom also has offices in Belgium, Taiwan and Hong Kong. For more information, visit www.ubicom.com.